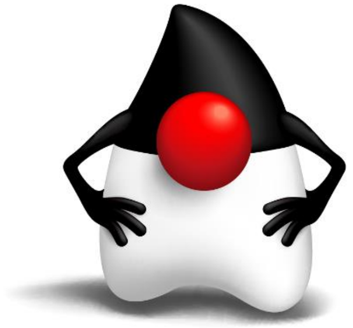
ORACLE®

JavaFX: Java's new Rich Client Platform



Java Pioneered Rich Client Applications

But developers had to learn multiple technologies

|

Tutorial and API Docs

http://docs.oracle.com/javase/8/javafx/get-started-tutorial/

http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

Ensemble – Collection of Examples

http://download.oracle.com/otndocs/products/javafx/2/samples/Ensemble/index.html

Tutorial and API Docs

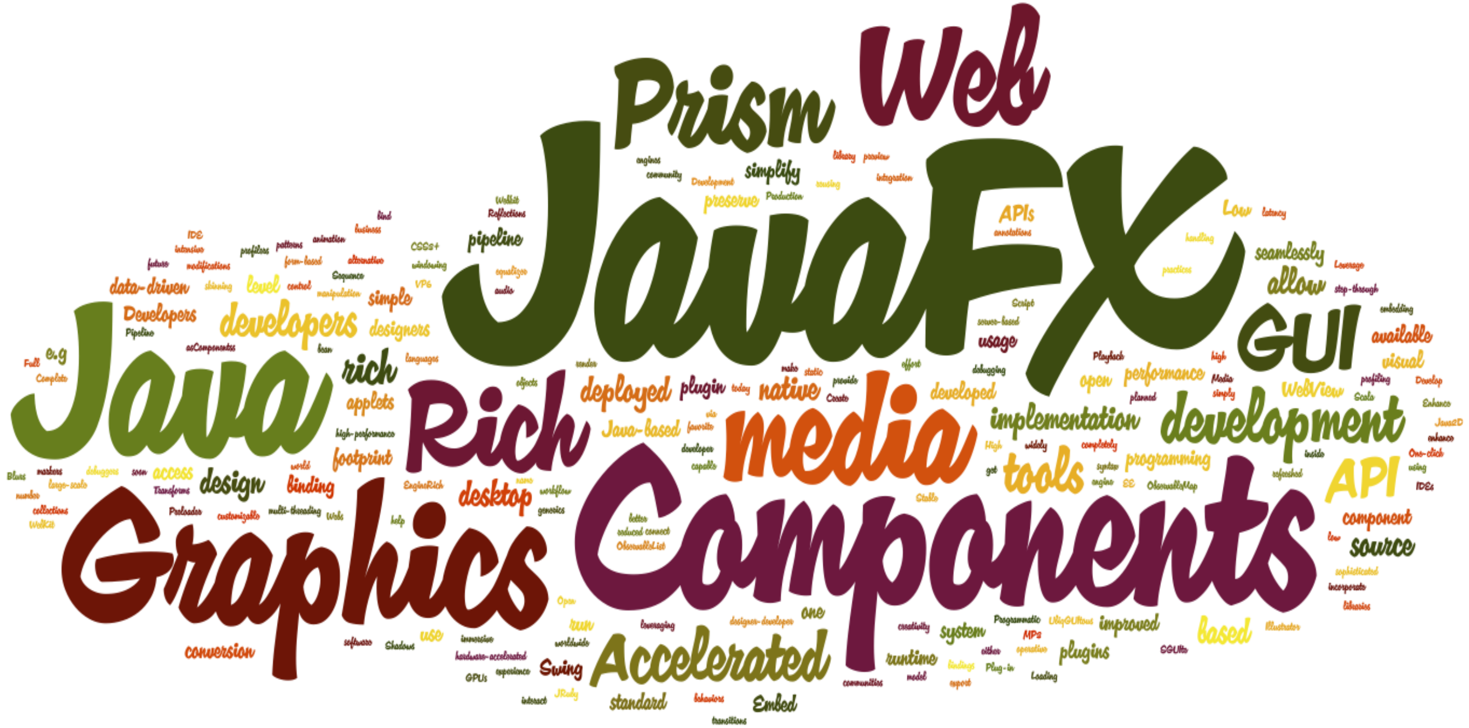
Videos on JavaFX
https://www.youtube.com/user/OracleLearning/search?query=javafx

How to create JavaFX project in Intellij:
https://www.jetbrains.com/help/idea/javafx.html

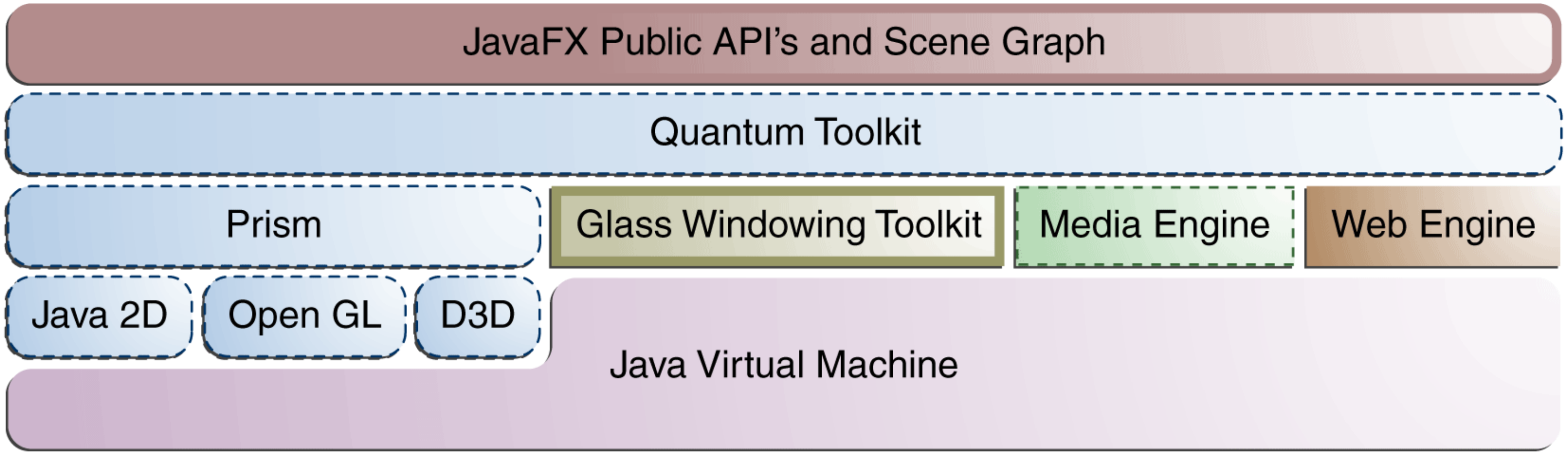
How to add SceneBuilder in Intellij:
https://www.jetbrains.com/help/idea/opening-fxml-files-in-javafx-scene-builder.html

JavaFX Simplifies Application Development

Developers Focus on Capabilities Instead of Technologies



JavaFX Runtime High Level Architecture



Threads in JavaFX

- **JavaFX application thread**: This is the primary thread used by JavaFX application developers. Any “live” scene, which is a scene that is part of a window, must be accessed from this thread. A scene graph can be created and manipulated in a background thread, but when its root node is attached to any live object in the scene, that scene graph must be accessed from the JavaFX application thread. This enables developers to create complex scene graphs on a background thread while keeping animations on 'live' scenes smooth and fast.
- **Prism render thread**
- **Media thread**

Java APIs and FXML

Java APIs for JavaFX	FXML
• End-to-end Java development • Java language features - generics, annotations, multi-threading • Fluent API for UI construction • Alternative JVM supported languages (e.g. Groovy, Scala) with JavaFX • Leverage sophisticated Java IDEs, debuggers and profilers • Java APIs preserve convenient JavaFX Script features (e.g., bind)	• Scriptable, XML-based markup language for defining UI • Convenient alternative to developing UI programmatically in Java • Easy to learn and intuitive for developers familiar with web technologies or other markup based UI technologies • Powerful scripting feature allows embedding scripts within FXML. Any JVM scripting language can be used, including JavaScript, Groovy, and Scala

WebView and Swing Interoperability

WebView Component

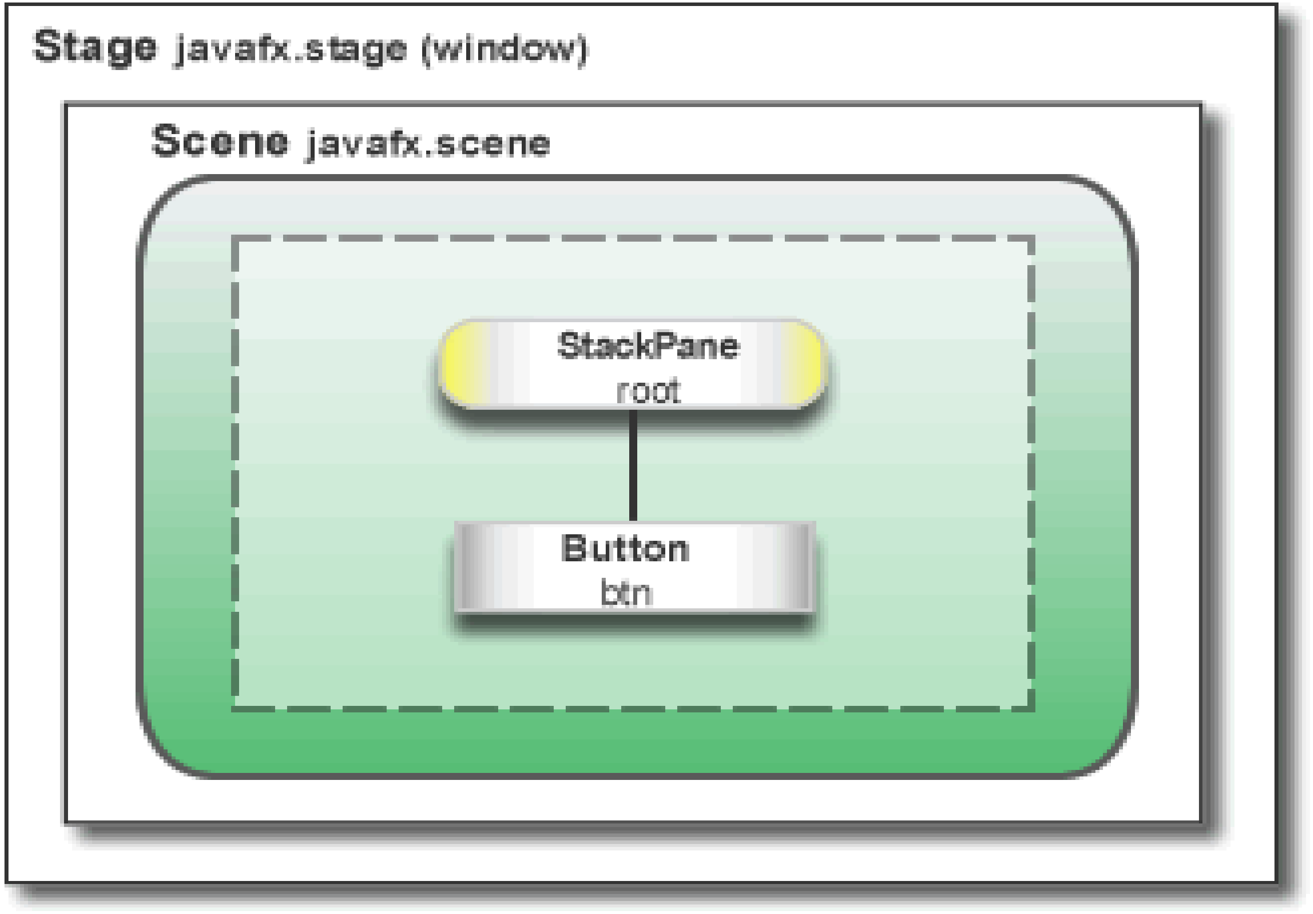
- Embed Web content in JavaFX applications
- HTML rendering based on Webkit
- Hardware accelerated rendering using PRISM
- DOM access and manipulation

Swing Interop

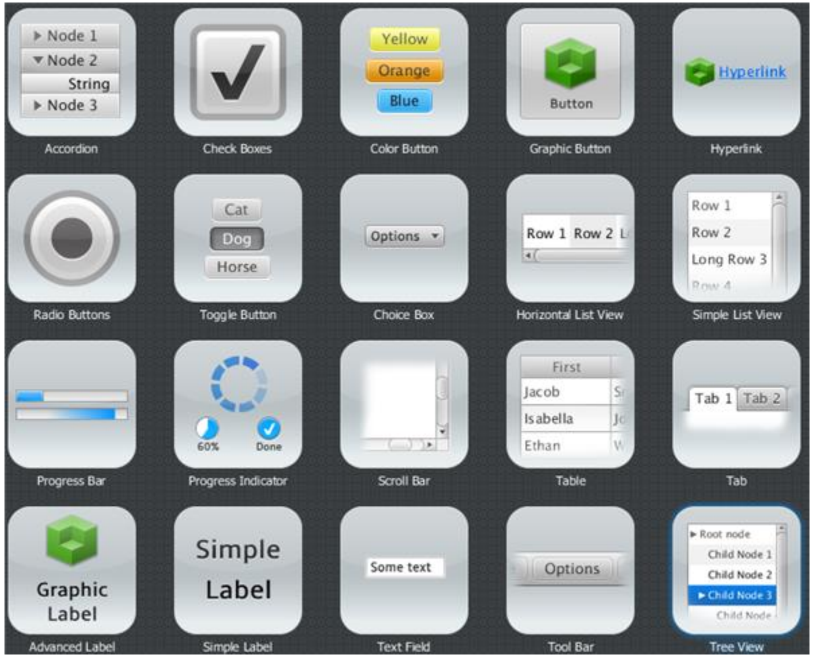
- Embed JavaFX content into existing Swing applications
- Extend existing Swing applications with new JavaFX features such as WebView and high-performance graphics

Browser Plugin

- Faster loading of JavaFX Web applications based on Prism
- Pre-loader for improved user experience with JavaFX Web applications

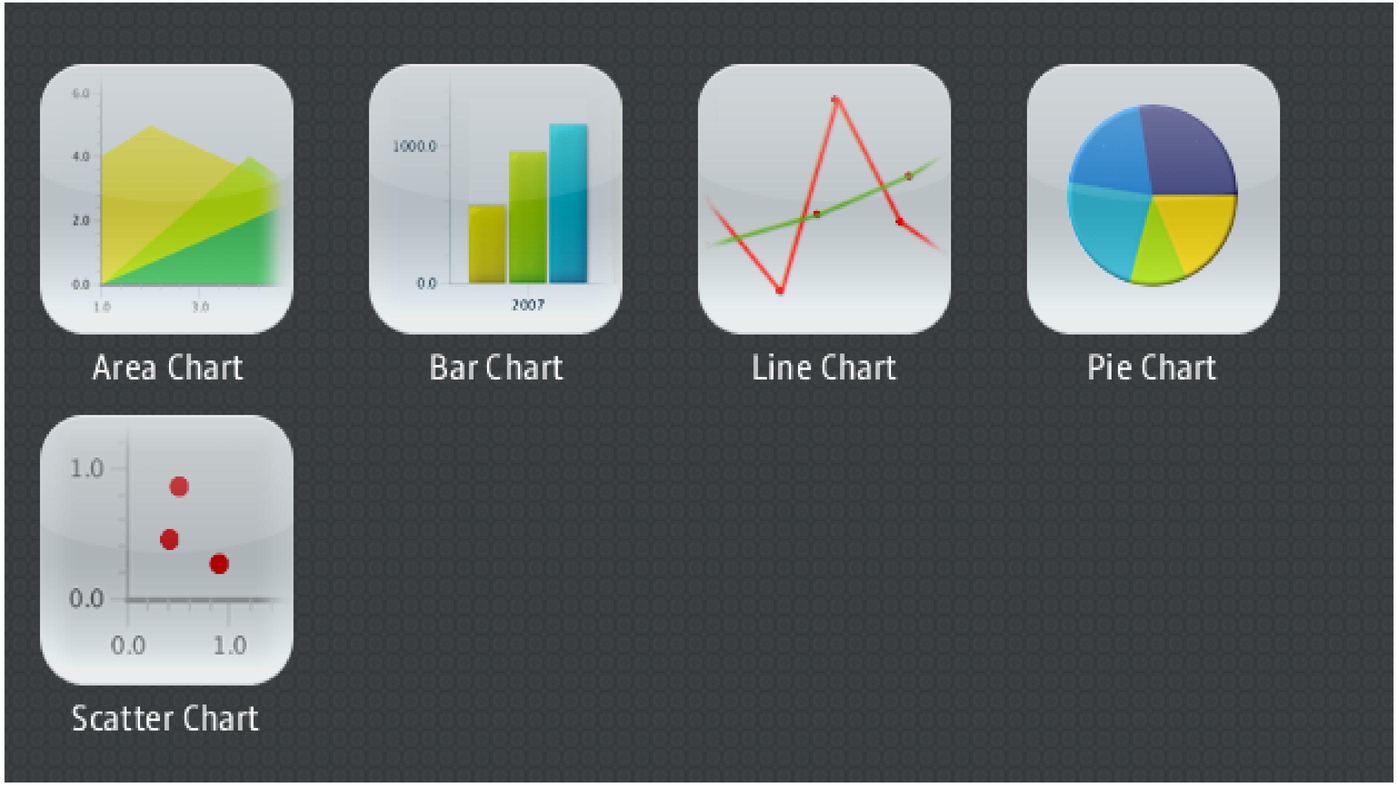

Stage javafx.stage (window)
Scene javafx.scene
StackPane
root
Button
btn

UI Controls

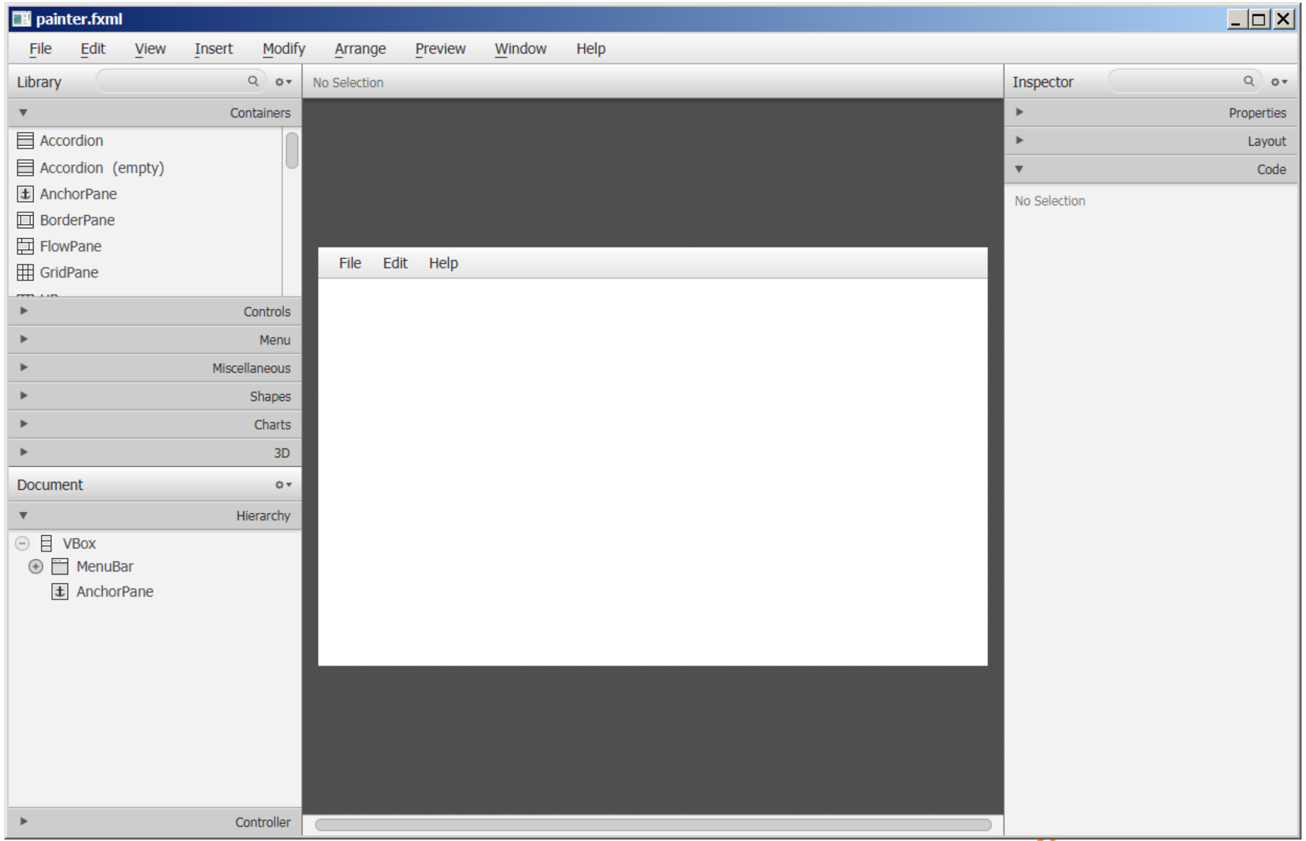


```
java --module-path "F:\javafx-sdk-11.0.2\lib" --add-modules
    javafx.controls,javafx.fxml -jar Ensemble8.jar
```

Charts

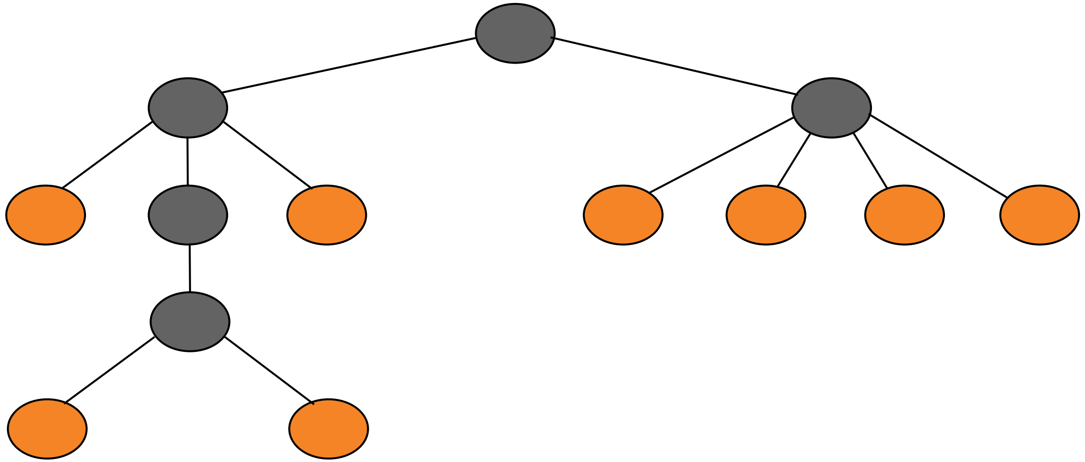

Area Chart
Bar Chart
Line Chart
Pie Chart
Scatter Chart
6.0
4.0
2.0
0.0
1.0
3.0
1000.0
0.0
2007
1.0
0.0
0.0
1.0

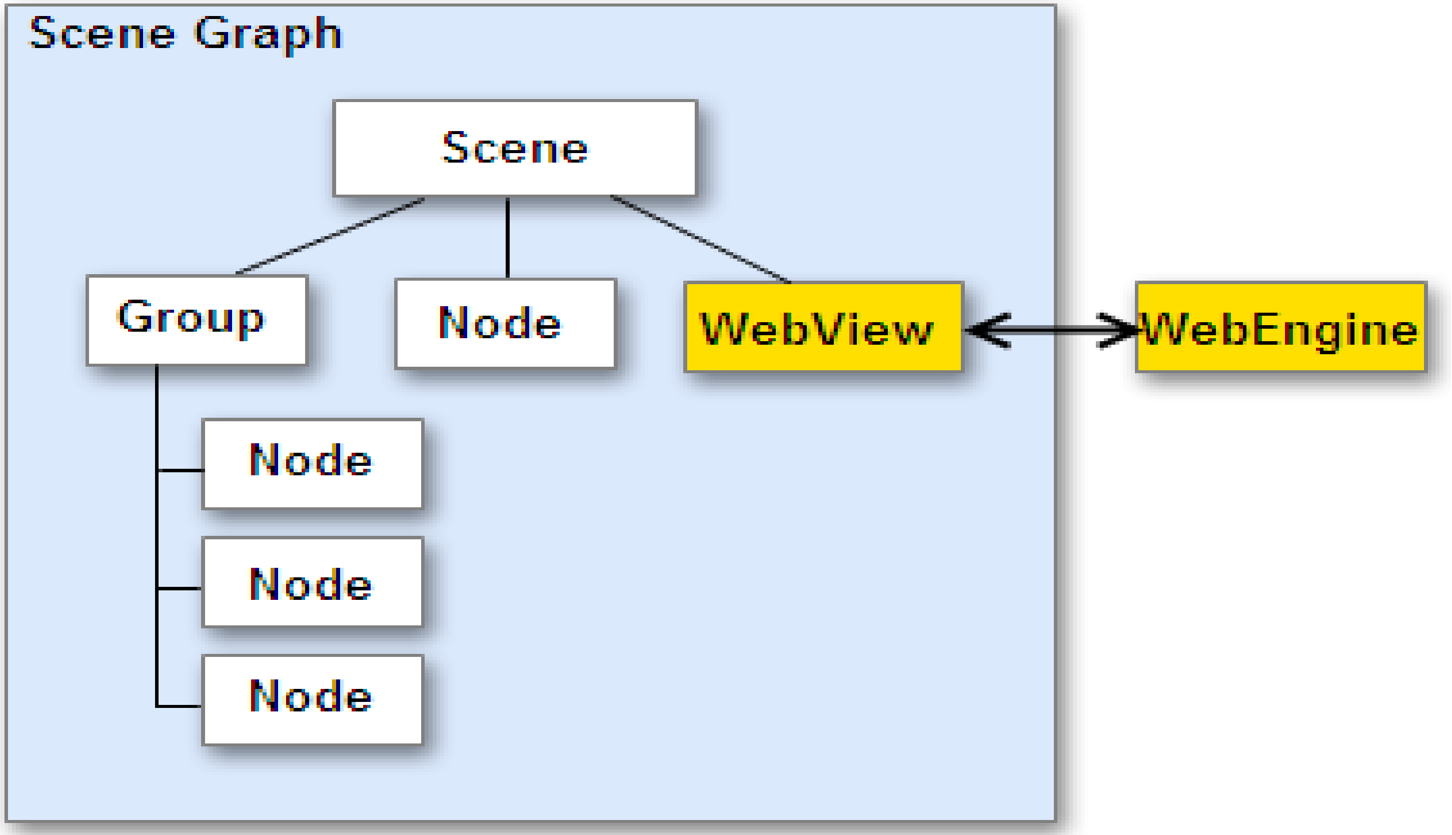
SceneBuilder



Scene Graph

- Directed Acyclic Graph
- Parents and children
- Representation of the GUI components




Scene Graph
Scene
Group
Node
WebView
WebEngine
Node
Node
Node

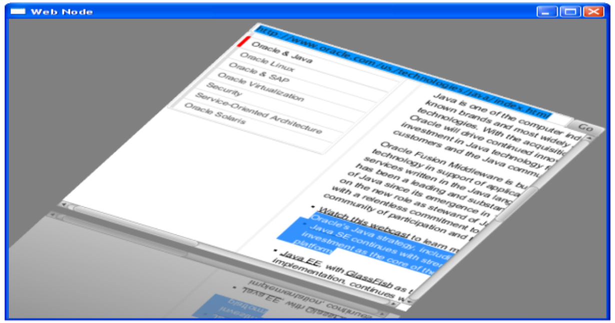
Media

- JavaFX supports both visual and audio media
- Cross-platform JavaFX media file format (fxm, mp3)
  - Platform specific formats supported via native players
- Media class represents a media file
- MediaPlayer provides control of the media rendering
- MediaView uses MediaPlayer to render media as Node
  - Many MediaViews can use the same MediaPlayer (cheaply)

Adding HTML Content

The Embedded Browser

- WebEngine
  - Provides basic web page browsing functionality
  - Supports user interaction: navigating links, submitting forms
- WebView
  - Web page as a Node in scenegraph
    - Effects can be applied
  - Encapsulates WebEngine object
  - No plugin support



Effects...

GaussianBlur

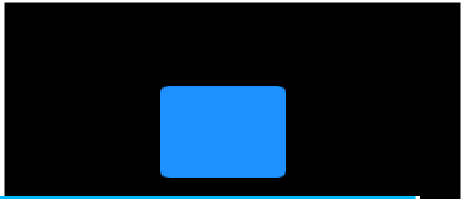
InnerShadow

Reflection

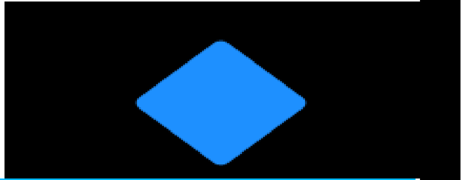
SepiaTone

Transforms

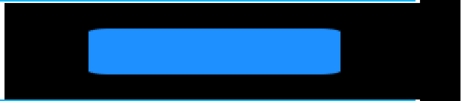
```
Rectangle rect=new Rectangle(0,0,60,60);
rect.setFill(Color.DODGERBLUE);
rect.setArcWidth(10);
rect.setArcHeight(10);
```



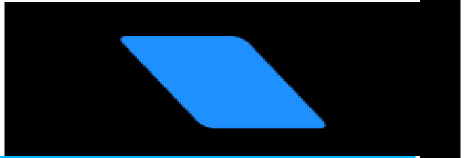
```
rect.setRotate(45);
```



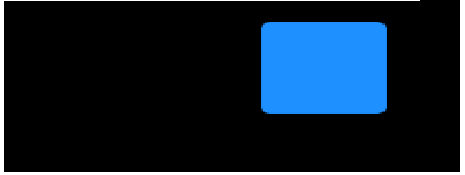
```
rect.setScaleX(2);
rect.setScaleY(0.5);
```



```
Shear shear = new Shear(0.7, 0);
rect.getTransforms().add(shear);
```

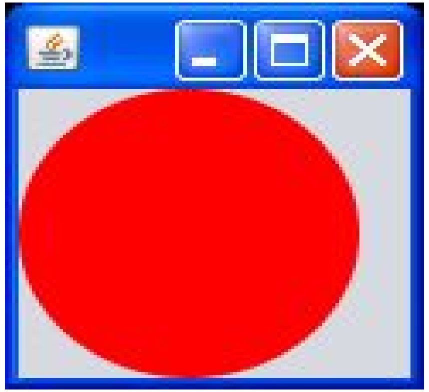


```
rect.setTranslateX(40);
rect.setTranslateY(10);
```



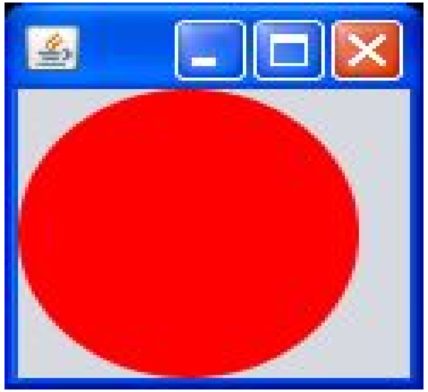
Let’s Compare: JavaFX 2.0

```
public class JavaFXTest extends Application {
  @Override public void start(Stage stage) {
    Group root = new Group();
    Scene scene = new Scene(root,100,100);
    stage.setScene(scene);

    Circle c1 =
      new Circle(50.0f, 50.0f, 50.0f, Color.RED);

    root.getChildren().add(c1);
    stage.setVisible(true);
  }

  public static void main(String a[]) {
    Launcher.launch(JavaFXTest.class, null);
  }
}
```



Let’s Compare: FXML

```
<BorderPane>
  <center>
    <Circle radius="50" centerX="50" centerY="50"/>
  </center>
</BorderPane>
```



```
public class JavaFXTest extends Application {
  @Override public void start(Stage stage) {
    stage.setTitle("FXML Example");
    Parent root = FXMLLoader.load(getClass().getResource("example.fxml"),
        ResourceBundle.getBundle("r.fxml_example"));
    stage.setScene(new Scene(root));
    stage.show();

  }
}
```

Binding

- Creates a dependency between a property and a changeable value
- High level API
  - Easy to use
  - Covers most common situations
- Low level API
  - Allows for more complex interactions
  - Optimised for fast execution and small footprint

Properties

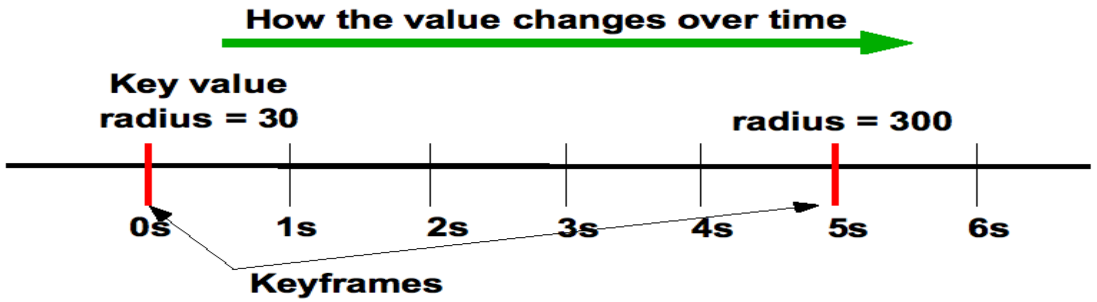
- Basis for high level binding API
- Concrete types for all primitives, String and Object
  - `DoubleProperty`, `StringProperty`, etc
- Simple API
  - `bind` / `unbind`
  - `bindBidirectional` / `unbindBidirectional`
  - `isBound`

Simple Binding Example

```
private SimpleDoubleProperty topXProperty =
  new SimpleDoubleProperty();
private SimpleDoubleProperty topYProperty =
  new SimpleDoubleProperty();

Line foldLine = new Line();
foldLine.setEndX(200);
foldLine.setEndY(200);
foldLine.startXProperty().bind(topXProperty);
foldLine.startYProperty().bind(topYProperty);

...

topXProperty.set(tx);
topYProperty.set(ty);
```

Timeline Based Animations

- **`Timeline`**
  - Modifies values of variables specified in KeyFrames
- **`KeyFrame`**: specifies that a variable should have
  - A particular value at a particular time
- **`KeyValue`**: Value to be interpolated for an interval



Animated Transitions

- Pre-defined, single-purpose animations
  - Fade, Path, Pause, Rotate, Scale, Translate
  - Can specify to, from and by values
- Container transitions
  - Parallel, sequential
  - Can be nested arbitarily
- Transitions and Timelines share ancestry
  - A Timeline can be added to a Parallel / Sequential transition

Standard Java Tools for Easy Development

- Source editor with improved syntactic highlighting, code completion, refactoring etc.
- Full debugger and profiler support
- Project wizard for easy creation of JavaFX applications

Other Java IDEs

- Source editor with syntactic highlighting, code completion, refactoring etc.
- Full debugger and Profiler support

JavaFX is …

- Cross platform: Windows GA, Mac & Linux Dev. Preview
- Familiar: 100% Java APIs
- Powerful: leverages underlying Java platform
- Modern: CSS skinning, HW acceleration, Webkit
- Backwards ‘compatible’: Swing & SWT interoperability
- Flexible: applicable to embedded, tablets and mobile
- Open Source: http://openjdk.java.net/projects/openjfx