JavaFX 8 Teil 2

CSS, Media-Player, Animations

CSS Stylesheets

- Stylesheets determine the look of the UI
- separate *.css file or inline definition
- Mostly known from HTML
- JavaFX CSS Reference:

http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

CSS Stylesheets

Adding a stylesheet

```
Scene scene = new Scene(new Group(), 500, 400);
scene.getStylesheets().add("path/stylesheet.css");
```

JavaFX Default CSS Stylesheet: modena.css

(extract using: jar xf jfxrt.jar com/sun/javafx/scene/control/skin/modena/modena.css)

CSS Stylesheets - Example

```
.root {
     -fx-background-color:#ccffcc;
}

.button {
     -fx-font: 16px "Sans-Serif";
     -fx-background-color:#99ff99;
     -fx-effect: dropshadow( one-pass-box , black , 8 , 0.0 ,
2 , 0 );
}

.label {
     -fx-font:25pt "Sans-Serif";
     -fx-padding: 10;
}
```

CSS Stylesheets - Selectors

Style classes:

.button
.check-box
.scroll-bar

...

Pseudo classes:

.radio-button:focused
.radio-button:hover
.scroll-bar:vertical

...

Descendant classes:

.check-box .label
.check-box .box
.radio-button .dot

...

Scene:

.root

CSS Stylesheets - Rule Properties

CSS property names correspond to the names of the properties for a class. Prefaced with “-fx-”

Some Examples:

-fx-font: 16px "Serif";
-fx-padding: 10;
-fx-background-color: #CCFF99;
-fx-background: rgb(225, 228, 203);
-fx-text-fill: white;
-fx-alignment: CENTER;
-fx-font-size: 16pt;
-fx-font-family: "Courier New";
-fx-base: rgb(132, 145, 47);

CSS Stylesheets - Class Styles

Within the stylesheet:

```
.button1 {
    (...)
}
```

use **styleClass**-attribute in FXML or like this code:

```
Button button = new Button("Click me!");
button.getStyleClass().add("button1");
```

CSS Stylesheets - ID Styles

Within the stylesheet:

```
#button-id {
    (...)
}
```

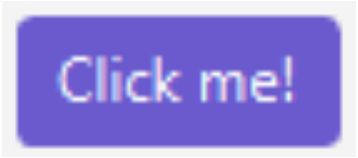
use with **id**-attribute in FXML or like this code:

```
Button button = new Button("Click me!");
button.setId("button-id");
```

Styling with CSS - Inline Style

CSS Stylings can be defined within your code:

```
Button button = new Button("Click me!");
button.setStyle("-fx-background-color:
slateblue; -fx-text-fill: white;");
```



Media Playback

The javafx.scene.media package allows for easy media playback of videos and audio based on the following three entities

Media Playback

Supported Media Codecs:

- **Audio:**
  MP3; AIFF containing uncompressed PCM; WAV containing uncompressed PCM; MPEG-4 multimedia container with Advanced Audio Coding (AAC) audio
- **Video:**
  FLV containing VP6 video and MP3 audio; MPEG-4 multimedia container with H.264/AVC (Advanced Video Coding) video compression

Media Playback - Media

- Represents media resource containing information like duration, metadata, tracks, and video resolution
- Instantiate with a string containing a URI to the audio/video file

```
Media media = new Media(mediaURI);
```

Media Playback - MediaPlayer

- The key component offering controls for playing media.
- Instantiate with a Media Object

```
MediaPlayer mediaPlayer = new MediaPlayer(media);
```

- provides methods that can be mapped to UI controls

```
play(), pause(), stop(), seek(), …
```

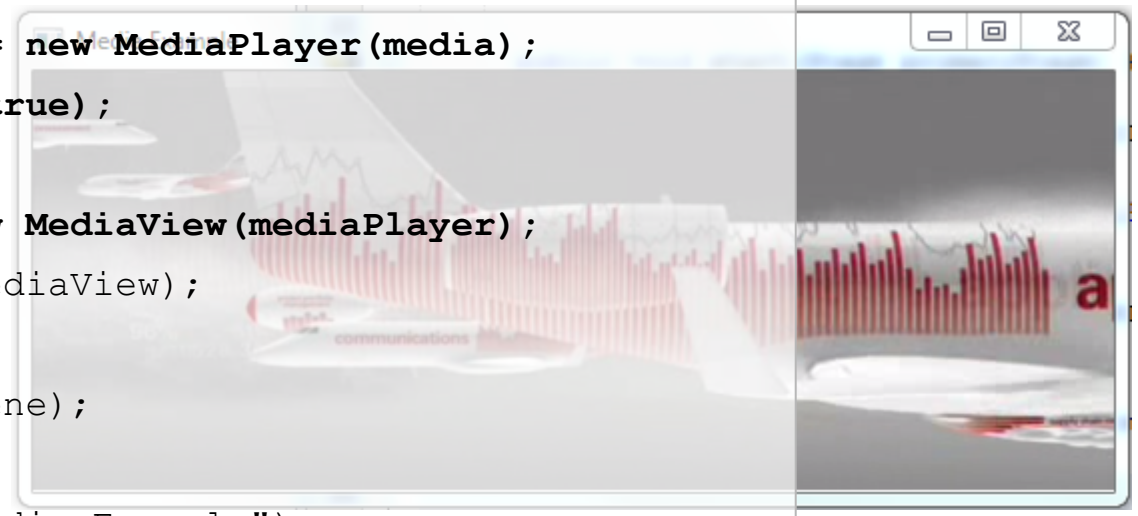
- Properties for:
  rate, autoPlay, balance, mute, volume, ...

Media Playback - MediaView

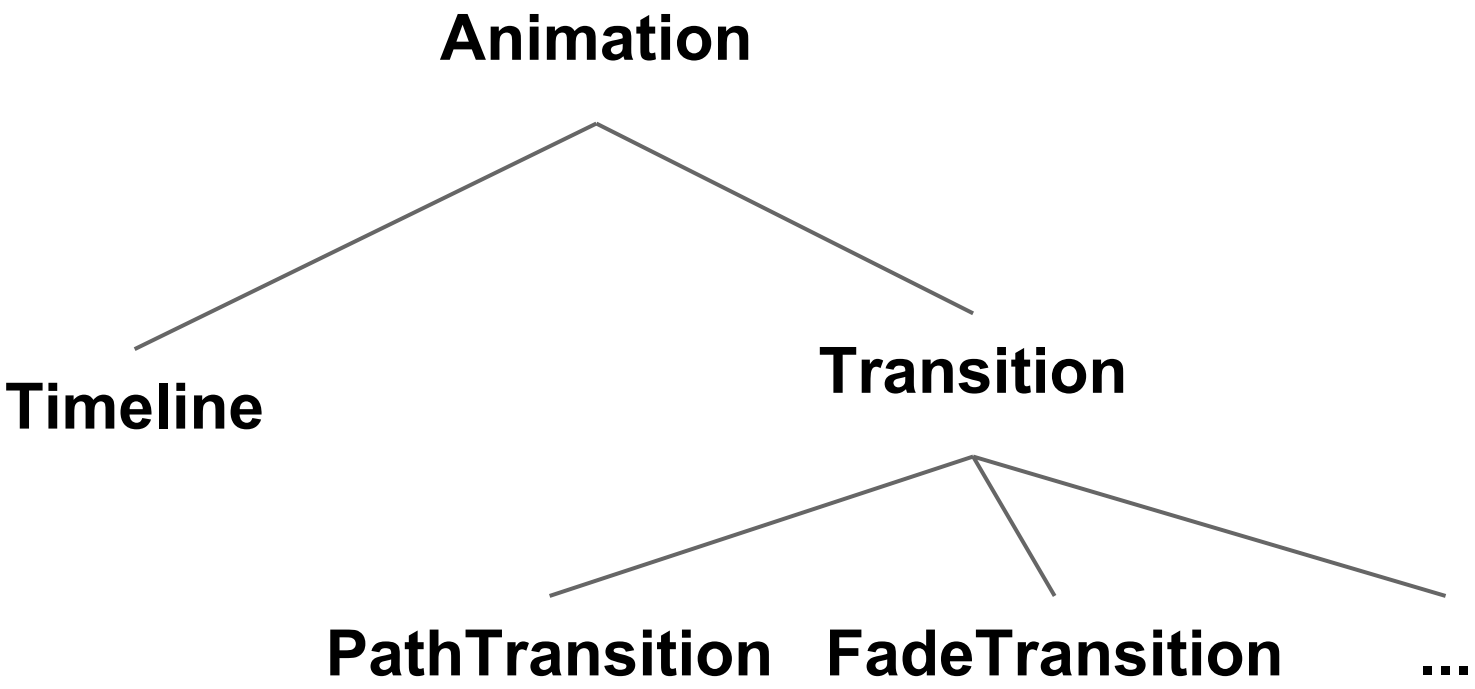
- A node object offering a view of the media being played with support for animation, translucency, and effects
- Instantiate with a MediaPlayer or set it later

```
MediaView mediaView = new MediaView(mediaPlayer); root.
getChildren().add(mediaView);
```

```
public class MediaExample extends Application {
    public void start(Stage primaryStage) {
    Group root = new Group();
    Scene scene = new Scene(root, 540, 210);

    String mediaURI = "http://download.oracle.
com/otndocs/products/javafx/oow2010-2.flv";
    Media media = new Media(mediaURI);

    MediaPlayer mediaPlayer = new MediaPlayer(media);
    mediaPlayer.setAutoPlay(true);

    MediaView mediaView = new MediaView(mediaPlayer);
    root.getChildren().add(mediaView);

    primaryStage.setScene(scene);
    primaryStage.show();
    primaryStage.setTitle("Media Example");
    }
(...)
```



Animations and Transitions



Animations

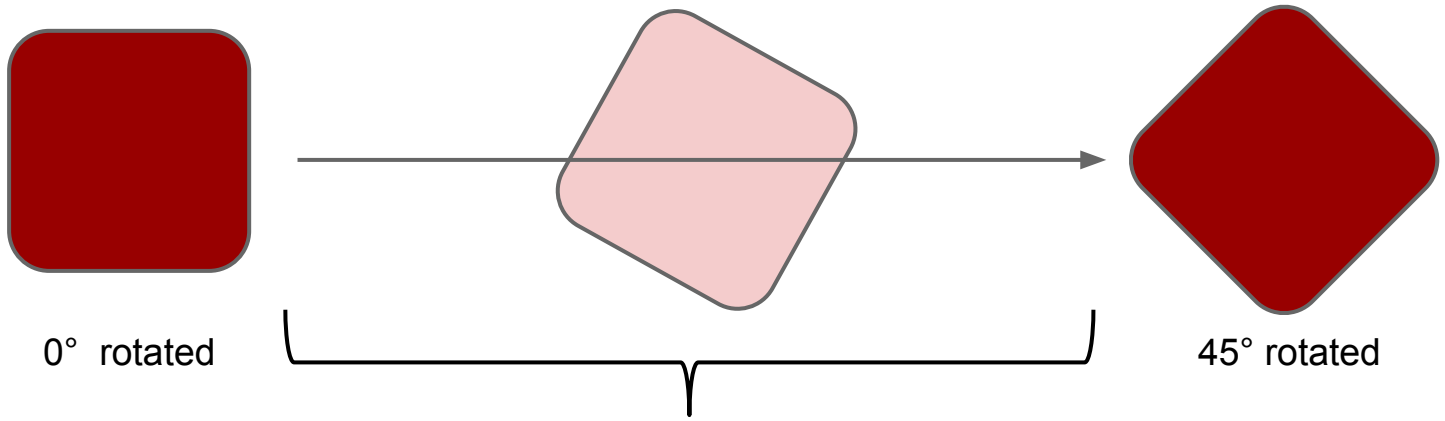
Can be applied to all JavaFX Nodes

Two Types of Animations in JavaFX:

Transition and **Timeline**

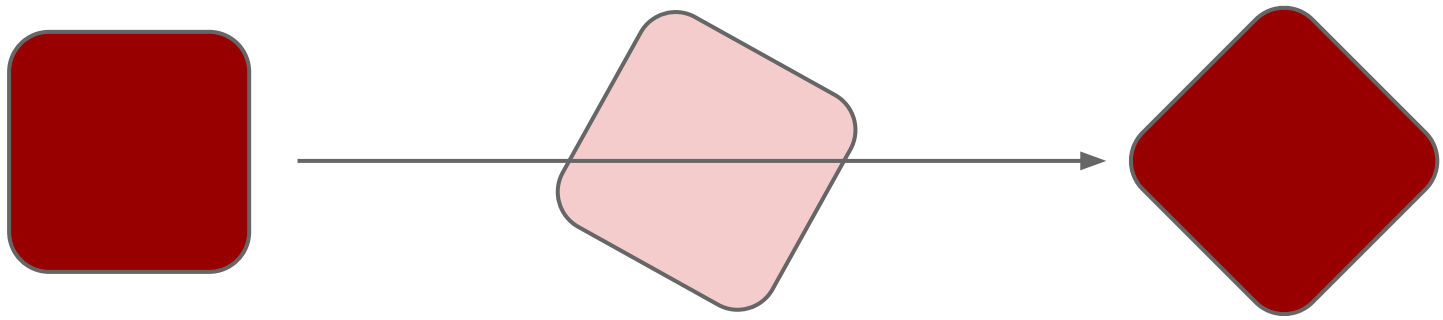
Transitions

Specify an Animation by declaring the **from-State** and **to-State** of a Property and the Duration in which the Animation should interpolate between them



Transitions

```
Rectangle rect = new Rectangle;
RotateTransition ft = new RotateTransition(Duration.millis(3000),rect);
ft.setFromValue(0);
ft.setToValue(45);
ft.play();
```



fromValue: 0°

toValue: 45°

Transitions

- Can be Combined
- Can be Eased with Interpolators
- You can even define your own Interpolator
- Can be strung together with a SequentialTransition
- Can be played in parallel with a ParallelTransition

Transition Types:

FadeTransition, FillTransition, PathTransition, TranslateTransition, RotateTransition, ScaleTranstion, StrokeTransition

Combine Transitions

Play transitions sequentially with **SequentialTransitions**:

```
SequentialTransition st;
st = new SequentialTransition(animationTarget,transition1,transition2);
```

...or in parallel with **ParallelTransitions**

```
ParallelTransition st;
st = new ParallelTransition(animationTarget,transition1,transition2);
```

Timeline Animation

Specify KeyFrames that have one or more KeyValues at a certain point in Time. Add these KeyFrames to a Timeline.

```
timeline = new Timeline();
Duration duration = Duration.millis(2000);
KeyValue keyValue = new KeyValue(animationTarget.scaleXProperty(), 2);
KeyFrame keyFrame = new KeyFrame(duration, keyValue)
timeline.getKeyFrames().add(keyFrame);
timeline.play();
```

Timeline Animation- additional features

Adding specific actions when each frame is started with the **AnimationTimer** class:

```
timer = new AnimationTimer() {

            @Override
            public void handle(long l) {
                //Do Something
            }
};

timeline.start();
timer.start();
```

Specify what happens when the Animation is finished with an EventHandler

```
EventHandler onFinished = new
EventHandler<ActionEvent>() {

            public void handle(ActionEvent t) {
                //Do Something
            }
};

KeyFrame keyFrame = new KeyFrame(duration,
onFinished,keyValue);
```

Interpolators

The Interpolator can be set with

setInterpolator(Interpolator value)

Predefined Interpolators

EASE_IN	**LINEAR**
EASE_OUT	**DISCRETE**
EASE_BOTH	

You can also define your own Interpolator with your own interpolate(double t) method

Links

More about Transitions and Animations:

http://docs.oracle.com/javase/8/javafx/visual-effects-tutorial/animations.htm#JFXTE149

JavaFX 8 API

http://docs.oracle.com/javase/8/javafx/api/index.html