# JavaFX

## Denne artikel beskriver Java's nye GUI framework JavaFX.

## Den forudsætter et vist kendskab til Java og GUI udvikling.

Skrevet den **28. jul 2013** af **arne_v** I kategorien **Programmering / Java** | ⭐⭐⭐⭐⭐

Historie:
V1.0 - 28/07/2013 - original

### Hvad er JavaFX

JavaFX er et nyt GUI framework til Java som skal erstatte Swing.

| Java version | År | GUI framework |
|---|---|---|
| 1.0-1.1 | 1996-1998 | AWT |
| 1.2-1.7 | 1998-2014 | Swing |
| 1.8- | 2014- | JavaFX |

### Status

JavaFX er standard i Java 8.

JavaFX er bundlet med Java 7 siden update 6.

For Java 6 og tidlige versioner af Java 7 kan man hente JavaFX som separat kit:
http://www.oracle.com/technetwork/java/javafx2-archive-download-1939373.html

### Versioner

JavaFX har en lidt broget historie.

JavaFX 1.0 blev releaset tilbage i 2008.

Men JavaFX 1.x er meget anderledes end den nuværende JavaFX 2.x.

JavaFX 1.x bestod af et library og et helt nye sprog kaldet JavaFX Script.

JavaFX Script var et meget spændende sprog med en syntax som kombinerede
normal imperativ programmings logik med en deklarativ definering af GUI layout.

JavaFX 2.0 blev releaset i 2011 og lavede om på meget.

Librariet blev udvidet.

JavaFX Script udgik. JavaFX kode skulle fremover udelukkende skrives i Java.

Et markup sprog til GUI layout FXML blev introduceret.

JavaFX 2.2.7 er versionen som er inkluderet i Java 7.

Hvis nogen er interesseret i JavaFX Script så lever det videre i 2 open source sprog:
* Visage @ GitHub
* F3 @ Google Code

Både SUN og Oracle har efter sigende brugt rigtigt mange millioner dollars på udviklingen af JavaFX.

**Filosofi**

Ideen i JavaFX 2.x er kopieret fra Adobe og Microsoft.

Adobe Flex introducerede i 2004 separeringen i markup og kode for GUI apps (MXML og ActionScript).

Microsoft kopierede ideen i 2006 med WPF (XAML og C#/VB.NET).

Så JavaFX Script fra JavaFX 1.x med den originale mixede imperative/deklarative kode blev droppet og erstattet af:
* FXML markup
* Java kode

For at gøre det nemt for traditionelle web udviklere at skifte til JavaFX GUI udvikling introducerede man at:
* styling sker ved hjælp af CSS
* kode også kan skrives i JavaScript

JavaFX er et meget stort framework. Hvis man kigger på Java 8 distribution så ser man at jfxrt.jar med JavaFX fylder 18 MB og rt.jar med stort set resten af Java library fylder 63MB.

**Simple eksempler**

Jeg vill illustrere JavaFX mulighederne med en lille GUI applikation.

Applikationen er banal og design inkl. valg farver er håbløs, men den har en passende kompleksitet til en demonstration.

Først en "business logic" klasse som bruges af GUI.

Calculator.java:

```
package demo;

public class Calculator {
    private double number;
    public void setNumber(double number) {
        this.number = number;
    }
```

```
    public double getSqrt() {
        return Math.sqrt(number);
    }
    public double getLog() {
        return Math.log(number);
    }
    public double getExp() {
        return Math.exp(number);
    }
}
```

Lad os starte med at vide en god gammeldags Swing GUI.

SwingGUI.java:

```
package demo;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class SwingGUI extends JFrame {
    private Calculator data = new Calculator();
    private JTextField number;
    private JLabel sqrt;
    private JLabel log;
    private JLabel exp;
    public SwingGUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Swing GUI");
        getContentPane().setLayout(new GridLayout(5, 2));
        getContentPane().setBackground(Color.YELLOW);
        getContentPane().add(new JLabel("Number"));
        number = new JTextField();
        getContentPane().add(number);
        getContentPane().add(new JLabel("Square root"));
        sqrt = new JLabel();
        sqrt.setForeground(Color.BLUE);
        getContentPane().add(sqrt);
        getContentPane().add(new JLabel("Logarithm"));
        log = new JLabel();
        log.setForeground(Color.BLUE);
        getContentPane().add(log);
        getContentPane().add(new JLabel("Exponential"));
        exp = new JLabel();
```

```
        exp.setForeground(Color.BLUE);
        getContentPane().add(exp);
        JButton calc = new JButton("Calculate");
        calc.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                double x = Double.parseDouble(number.getText());
                data.setNumber(x);
                sqrt.setText(Double.toString(data.getSqrt()));
                log.setText(Double.toString(data.getLog()));
                exp.setText(Double.toString(data.getExp()));
            }
        });
        calc.setBackground(Color.RED);
        getContentPane().add(calc);
        pack();
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame f = new SwingGUI();
                f.setVisible(true);
            }
        });
    }
}
```

Det virker fint, men:
* der er meget "overflødig" kode
* GUI look & feel virker gammeldags

Nu prøver vi at lave den samme kode i JavaFX uden brug af FXML.

OldStyleJavaFX.java:

```
package demo;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class OldStyleJavaFXGUI extends Application {
    private Calculator data = new Calculator();
    private TextField number;
    private Label sqrt;
    private Label log;
    private Label exp;
    @Override
```

```
    public void start(Stage stg) throws Exception {
        stg.setTitle("Old style JavaFX GUI");
        GridPane pn = new GridPane();
        pn.setStyle("-fx-background-color: yellow;");
        number = new TextField();
        pn.add(new Label("Number"), 0, 0);
        pn.add(number, 1, 0);
        pn.add(new Label("Square root"), 0, 1);
        sqrt = new Label();
        sqrt.setStyle("-fx-text-fill: blue;");
        pn.add(sqrt, 1, 1);
        pn.add(new Label("Logarithm"), 0, 2);
        log = new Label();
        log.setStyle("-fx-text-fill: blue;\"");
        pn.add(log, 1, 2);
        pn.add(new Label("Exponential"), 0, 3);
        exp = new Label("");
        exp.setStyle("-fx-text-fill: blue;\"");
        pn.add(exp, 1, 3);
        Button calc = new Button("Calculate");
        calc.setStyle("-fx-background-color: red;");
        calc.setOnAction((e) -> {
            double x = Double.parseDouble(number.getText());
            data.setNumber(x);
            sqrt.setText(Double.toString(data.getSqrt()));
            log.setText(Double.toString(data.getLog()));
            exp.setText(Double.toString(data.getExp()));
        });
        pn.add(calc, 0, 4);
        Scene scn = new Scene(pn);
        stg.setScene(scn);
        stg.show();
    }
    public static void main(String[] args) {
        Application.launch(OldStyleJavaFXGUI.class, args);
    }
}
```

Koden er en anelse pænere men ikke meget.

GUI look & feel virker dog mere moderne.

Nu prøver vi så at flytte layout og styling ud i FXML og CSS.

TrueJavaFXGUI.java:

```
package demo;

import javafx.application.Application;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class TrueJavaFXGUI extends Application {
    private Calculator data = new Calculator();
    @FXML
    private TextField number;
    @FXML
    private Label sqrt;
    @FXML
    private Label log;
    @FXML
    private Label exp;
    @Override
    public void start(Stage stg) throws Exception {
        stg.setTitle("True JavaFX GUI");
        Scene scn = new Scene((Parent)
FXMLLoader.load(getClass().getResource("JavaFXGUI.fxml")));

scn.getStylesheets().add(getClass().getResource("JavaFXGUI.css").toString());
        stg.setScene(scn);
        stg.show();
    }
    public void calcClick() {
        double x = Double.parseDouble(number.getText());
        data.setNumber(x);
        sqrt.setText(Double.toString(data.getSqrt()));
        log.setText(Double.toString(data.getLog()));
        exp.setText(Double.toString(data.getExp()));
    }
    public static void main(String[] args) {
        Application.launch(TrueJavaFXGUI.class, args);
    }
}
```

JavaFXGUI.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane xmlns:fx="http://javafx.com/fxml"
fx:controller="demo.TrueJavaFXGUI">
    <Label GridPane.columnIndex="0" GridPane.rowIndex="0" text="Number"/>
    <TextField GridPane.columnIndex="1" GridPane.rowIndex="0" fx:id="number"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="1" text="Square root"/>
```

```
    <Label GridPane.columnIndex="1" GridPane.rowIndex="1" fx:id="sqrt"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="2" text="Logarithm"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="2" fx:id="log"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="3" text="Exponential"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="3" fx:id="exp"/>
    <Button GridPane.columnIndex="0" GridPane.rowIndex="4" text="Calculate"
fx:id="calc" onAction="#calcClick" />
</GridPane>
```

JavaFXGUI.css:

```
.root {
    -fx-background-color: yellow;
}

.button {
    -fx-background-color: red;
}

#sqrt, #log, #exp {
    -fx-text-fill: blue;
}
```

Og nu begynder det at ligne noget. Kode, layout og styling er klart adskilt.

Hvis man vil kan man også flytte click event koden fra Java til JavaScript.

Først inline i FXML'en.

TrueJavaFXGUIwithJS.java:

```
package demo;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class TrueJavaFXGUIwithJS extends Application {
    private Calculator data = new Calculator();
    @Override
    public void start(Stage stg) throws Exception {
        stg.setTitle("True JavaFX GUI with JavaScript");
        Scene scn = new Scene((Parent)
FXMLLoader.load(getClass().getResource("JavaFXGUIwithJS.fxml")));

scn.getStylesheets().add(getClass().getResource("JavaFXGUI.css").toString());
```

```
        stg.setScene(scn);
        stg.show();
    }
    public Calculator getData() {
        return data;
    }
    public static void main(String[] args) {
        Application.launch(TrueJavaFXGUIwithJS.class, args);
    }
}
```

JavaFXGUIwithJS.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?language javascript?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane xmlns:fx="http://javafx.com/fxml"
fx:controller="demo.TrueJavaFXGUIwithJS">
    <fx:script>
        function calcClick() {
            x = parseFloat(number.text);
            controller.data.number = x;
            sqrt.text = controller.data.sqrt.toString();
            log.text = controller.data.log.toString();
            exp.text = controller.data.exp.toString();
        }
    </fx:script>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="0" text="Number"/>
    <TextField GridPane.columnIndex="1" GridPane.rowIndex="0" fx:id="number"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="1" text="Square root"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="1" fx:id="sqrt"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="2" text="Logarithm"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="2" fx:id="log"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="3" text="Exponential"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="3" fx:id="exp"/>
    <Button GridPane.columnIndex="0" GridPane.rowIndex="4" text="Calculate"
fx:id="calc" onAction="calcClick()" />
</GridPane>
```

Så med ekstern JS fil.

TrueJavaFXGUIwithExtJS.java:

```
package demo;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class TrueJavaFXGUIwithExtJS extends Application {
    private Calculator data = new Calculator();
    @Override
    public void start(Stage stg) throws Exception {
        stg.setTitle("True JavaFX GUI with external JavaScript");
        Scene scn = new Scene((Parent)
FXMLLoader.load(getClass().getResource("JavaFXGUIwithExtJS.fxml")));

scn.getStylesheets().add(getClass().getResource("JavaFXGUI.css").toString());
        stg.setScene(scn);
        stg.show();
    }
    public Calculator getData() {
        return data;
    }
    public static void main(String[] args) {
        Application.launch(TrueJavaFXGUIwithExtJS.class, args);
    }
}
```

JavaFXGUIwithExtJS.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?language javascript?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane xmlns:fx="http://javafx.com/fxml"
fx:controller="demo.TrueJavaFXGUIwithJS">
    <fx:script source="JavaFXGUI.js"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="0" text="Number"/>
    <TextField GridPane.columnIndex="1" GridPane.rowIndex="0" fx:id="number"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="1" text="Square root"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="1" fx:id="sqrt"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="2" text="Logarithm"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="2" fx:id="log"/>
    <Label GridPane.columnIndex="0" GridPane.rowIndex="3" text="Exponential"/>
    <Label GridPane.columnIndex="1" GridPane.rowIndex="3" fx:id="exp"/>
    <Button GridPane.columnIndex="0" GridPane.rowIndex="4" text="Calculate"
fx:id="calc" onAction="calcClick()" />
```

```
</GridPane>
```

JavaFXGUI.js:

```
function calcClick() {
    x = parseFloat(number.text);
    controller.data.number = x;
    sqrt.text = controller.data.sqrt.toString();
    log.text = controller.data.log.toString();
    exp.text = controller.data.exp.toString();
}
```

Bemærk at JavaScript eksemplerne ikke virker i Java 8 early access. De virker i Java 7 og de bør også virke i Java 8 final version.

**Browser & medie**

JavaFX har en control for en embedded browser. Browser bygger på WebKit (samme som bruges af Chrome, iPhone browser og Android browser).

JavaFXBrowser.java:

```
package demo;

import javafx.application.Application;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.web.WebView;
import javafx.stage.Stage;

public class JavaFXBrowser extends Application {
    @FXML
    private WebView brws;
    @Override
    public void start(Stage stg) throws Exception {
        stg.setTitle("JavaFX with embedded browser");
        Scene scn = new Scene((Parent)
FXMLLoader.load(getClass().getResource("JavaFXBrowser.fxml")));
        stg.setScene(scn);
        stg.show();
    }
    public void eksperten() {
        brws.getEngine().load("http://www.eksperten.dk/");
    }
    public void google() {
        brws.getEngine().load("http://www.google.com/");
```

```
    }
    public static void main(String[] args) {
        Application.launch(JavaFXBrowser.class, args);
    }
}
```

JavaFXBrowser.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.web.*?>

<BorderPane xmlns:fx="http://javafx.com/fxml"
fx:controller="demo.JavaFXBrowser">
    <center>
        <WebView fx:id="brws"/>
    </center>
    <bottom>
        <GridPane >
            <Button GridPane.columnIndex="0" GridPane.rowIndex="0" text="Go to
eksperten.dk" onAction="#eksperten" />
            <Button GridPane.columnIndex="1" GridPane.rowIndex="0" text="Go to
Google" onAction="#google" />
        </GridPane>
    </bottom>
</BorderPane>
```

JavaFX har en control for embedded media player.

JavaFXPlayer.java:

```
package demo;

import javafx.application.Application;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;
```

```
public class JavaFXPlayer extends Application {
    @FXML
    private MediaView media;
    @Override
    public void start(Stage stg) throws Exception {
        stg.setTitle("JavaFX with embedded player");
        Scene scn = new Scene((Parent)
FXMLLoader.load(getClass().getResource("JavaFXPlayer.fxml")), 800, 600);
        stg.setScene(scn);
        stg.show();
    }
    public void video() {
        media.setMediaPlayer(new MediaPlayer(new
Media("file:///c:/work/test.flv")));
        media.setFitWidth(2*360);
        media.setFitHeight(2*288);
        media.getMediaPlayer().setAutoPlay(true);
    }
    public void audio() {
        media.setMediaPlayer(new MediaPlayer(new
Media("file:///c:/work/test.mp3")));
        media.getMediaPlayer().setAutoPlay(true);
    }
    public static void main(String[] args) {
        Application.launch(JavaFXPlayer.class, args);
    }
}
```

JavaFXPlayer.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.media.*?>

<BorderPane xmlns:fx="http://javafx.com/fxml"
fx:controller="demo.JavaFXPlayer">
    <center>
        <MediaView fx:id="media"/>
    </center>
    <bottom>
        <GridPane >
            <Button GridPane.columnIndex="0" GridPane.rowIndex="0" text="Video
sample" onAction="#video" />
            <Button GridPane.columnIndex="1" GridPane.rowIndex="0" text="Audio
sample" onAction="#audio" />
        </GridPane>
    </bottom>
</BorderPane>
```

Bemærk at understøttede video og audio formater er ret begrænset.

**Swing & JavaFX**

Man kan bruge JavaFX komponenter i en Swing applikation.

Men man kan ikke bruge Swing komponenter i en JavaFX applikation i JavaFX 2.x (man kunne i JavaFX 1.x og der går rygter om at Oracle vil indføre muligheden igen senere).

MixedGUI.java:

```
package demo;

import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MixedGUI extends JFrame {
    public MixedGUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Mixed GUI");
        getContentPane().setLayout(new GridLayout(2, 1));
        JButton swinghi = new JButton("Swing hi");
        JLabel swingtxt = new JLabel("
         ");
        swinghi.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                swingtxt.setText("Hi from Swing");
            }
        });
        JPanel p = new JPanel();
        p.setLayout(new BorderLayout());
        p.add(swinghi, BorderLayout.WEST);
        p.add(swingtxt, BorderLayout.EAST);
        getContentPane().add(p);
        JFXPanel fxp = new JFXPanel();
```

```
        getContentPane().add(fxp);
        pack();
        Platform.runLater(() -> {
            Button javafxhi = new Button("JavaFX hi");
            Label javafxtxt = new Label("
           ");
            javafxhi.setOnAction((e) -> {
                javafxtxt.setText("Hi from JavaFX");
            });
            BorderPane bp = new BorderPane();
            bp.setLeft(javafxhi);
            bp.setRight(javafxtxt);
            Scene scn = new Scene(bp);
            fxp.setScene(scn);
        });
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame f = new MixedGUI();
                f.setVisible(true);
            }
        });
    }
}
```

Jeg vil frarråde at forsøge at blande Swing og JavaFX.

Forskellige look & feel, 2 event tråde etc. vil give masser af problemer.

**Applets**

Man kan også udvikle applets med JavaFX.

**Fremtid**

Som det fremgår af de foregående afsnit, så er JavaFX er meget spændende teknologi og et markant fremskridt i forhold til Swing.

Imidlertid tror jeg ikke at JavaFX bliver en success.

Java applets har været anset som forældet i forhold til Flash og SilverLight i mange år p.g.a. Swing. Nu har JavaFX så bragt Java op på niveau med disse, men alle 3 er nu forældede p.g.a. HTML 5 og manglende support i mobil browsere.

Java desktop apps har aldrig været en stor success. Og selvom JavaFX vil gøre Java desktop apps bedre og nemmere at udvikle så er det usandsynligt at Java for alvor skulel få fat i det marked.

For smartphone apps har Google valgt at give Android sit eget GUI framework, så heller ikke her vil JavaFX kunne finde en god niche.

Men for dem som har lyst til eller brug for at lave en desktop applikation:
* hvor samme binary vil køre på Windows, MacOS X, Linux pg Solaris
* med et moderne GUI framework
* ikke er bange for at gå nye veje
så er JavaFX absolut en interessant mulighed.

**Kommentar af acore d. 31. jul 2013 | 1**

Meget illustrativ og lærerig guide - ville ønske at der var noget mere af den karat.

Og får så helt lyst til at prøve JavaFX - har aldrig været rigtig god ven med Swing, men kan godt lide Java.