# Wieso wir JavaFX brauchen und was das überhaupt ist.

Niklas von Hirschfeld, Jan Knüpfer

2024-01-23

## Contents

## Was ist JavaFX?

JavaFX ist eine Bibliothek für Java, welche erstmals 2008 von Sun Microsystems veröffentlicht wurde. Es sollte ursprünglich eine Alternative zu Swing und AWT bieten, insbesondere im Hinblick auf die Entwicklung von Rich Internet Applications (RIAs) und ansprechenden Benutzeroberflächen. Es wird seit dem Weiterentwickelt und auch nach der Übernahme von Sun Microsystems durch Oracle hat sich daran nichts geändert. Es bietet neben bindings für OpenGL eine recht hohe API um auf diese zuzugreifen und ist ein Moderner nachfolger von dem veralteten Swing framework.

JavaFX nutzt einen *scenen Graf* um Objekte und Scenen zu sortieren und zugänglich.

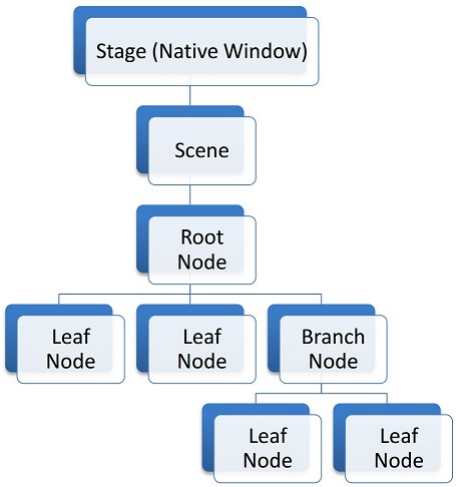


Figure 1: Grafisch darstellung einer *Stage*

Die Basis stellt eine *Stage* dar. Diese ist vergleichbar mit einer Bühne eines Theaterstückes, auf der Verschiedene Scenen und Schauspiele auftreten können. Eine Scene beinhaltet die Schauspieler und die Requisieten. In Java wären diese klassisches Objekte, welche durch einfache Formen oder Grafiken dargestellt werden. Diese Objekte beinhalten die Grafische darstellung, position, variablen und funktionen, welche für dieses Objekt wichtig sind.

## Was bietet JavaFX

Neben einer hohen und einfachen API zu OpenGL bietet JavaFX von Haus aus Module für *Text*, *Knöpfe* und Bilder. Es ermöglicht auch die einfache Verwendung von anderen Medien, wie Video- und Audiodatein.

```
public class HelloApp extends Application {

    private Parent createContent() {
        return new StackPane(new Text("Hello World"));
    }

    @Override
    public void start(Stage stage) throws Exception {
        stage.setScene(new Scene(createContent(), 300, 300));
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Diese Zeilen allein (plus Import der Libary) reichen aus, um ein Fenster mit dem Text "Hello World" zu erstellen. Anhand des Beispiels lässt sich auch die grundlegende Herangehensweise erkennen. Die Klasse `HelloApp` erweiter die Klasse `Application`, welche von JavaFX gestellt wird. Diese Klasse beinhaltet bereits eine reihe von Funktionen, welche genutzt oder auch überschrieben werden können. `start(Stage stage)` ist eine solche Funktion. Sie wird ausgeführt, nachdem JavaFX initialisiert wurde und stellt den Ort da, an dem die `stage` standardmäßig definiert wird. `privat Parent createContent()` stellt ein Objekt beziehungsweise in dem Fall nur eine einfache Funktion dar, welche eine `StackPane` mit einem Text zurück gibt. `stage.setScene(new Scene(...))` setzt die Szene mit. Das Praktische ist, Szenen können unabhängig und im vorraus erstellt werden.

## CSS

JavaFX nutzt eine eigene Version von `Cascading Style Sheets` (CSS), welches auf der W3C CSS Version 2.1 basiert. Dies wird in erster Linie genutzt, um Themen zu erstellen und zu bearbeiten.

JavaFX CSS unterstützt allerdings keine Layout Eigenschaften wie `float`, `position`, `overflow`, etc. Es gibt insgesamt viele Einschränkungen aber eben auch Erweiterungen, speziell für JavaFX.

## Event Handlers

In JavaFX werden Event Handler verwendet, um auf Benutzerinteraktionen oder andere ereignisbezogene Aktivitäten zu reagieren.

JavaFX bietet dabei Event-Klassen, zum Beispiel: MouseEvent, KeyEvent, ActionEvent und Event-Quellen, also JavaFX-Steuerelemente wie `Textfeld`, `Knöpfe`, etc.

Auf solche Event kann man mit `EventHandler` reagieren. Diese sind wie folgt aufgebaut:

```
EventHandler<ActionEvent> buttonClickHandler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        // Hier steht der Code, der auf den Klick der Schaltfläche reagiert
        System.out.println("Button wurde geklickt!");
    }
};
```

Wenn der `EventHandler` einmal definiert ist, kann dieser Verschiedenen Objekten hinzugefügt werden:

```
Button myButton = new Button("Klick mich!");
myButton.setOnAction(buttonClickHandler);
```

Man muss allerdings nicht im Voraus einen `EventHandler` an sich Definieren, sonder kann auch individuell für ein Objekt direkt beschreiben, was passieren soll:

```
myButton.setOnAction(e -> {
    System.out.println("Button wurde geklickt!");
});
```

## Module

### javafx.base

`module javafx.base` definiert die basis der API mit Bindings, Eigenschaften, Sammlungen und Ereignissen.

### javafx.controls

In diesem Modul sind die Elemente für eine Benutzeroberfläche, wie Knöpfe, Textfelder, Liste und noch viele andere Steuerelemente. Es beinhaltet ebenfalls Methoden um Layouts und die Anordnung der Elemente zu Managen.

### javafx.fxml

FXML ist ein ist ein Modul von JavaFX. Diese erlaubt es, Szenen als .xml Datein zu speichern. Dadurch ist es auch möglich, auf grafische Programme zurückzugreifen, um ein Nutzerinterface zu erstellen.

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8"
fx:controller="org.scenebuilder.BasicFXMLController">
<children>
<Button fx:id="button" layoutX="126.0" layoutY="90"
onAction="#handleButtonAction" text="Click Me!" />

<Label fx:id="label" layoutX="126" layoutY="120" minHeight="16"
minWidth="69" />
</children>
</AnchorPane>
```

### javafx.graphics

Dieses Modul ist für alle Grafik basierten Aspekte im Spiel zuständig. Darunter bietet es Funktionen zum Arbeiten mir Bildern, Formen und Farben. Es ist notwendig um jegliche grafischen Elemente zu Rendern.

### javafx.media

Für die Wiedergabe von Sound und Musik bietet dieses Modul hilfreiche und notwendige Funktionen und Klassen.

**javafx.swing**

Java Swing ist ebenfalls ein GUI-Toolkit welches seit 1998 Bestandteil der Java-Runtime ist. Abstract Window Toolkit (AWT) ist ein Teil von Swing. Dieses Modul ermöglicht eine beidseitige Kompatibilität und erlaubt so einen fließenden Übergang zu JavaFX, da Swing und AWT nicht mehr Weiterentwickelt werden. Swing und AWT werden zwar weiterhin unterstützt und gelten auch als stabil, es wird aber an keinen neuen Funktionen mehr gearbeitet.

**javafx.web**

Das Modul ermöglicht das Anzeigen von HTML-Seiten innerhalb der JavaFX-Anwendungen. Dadurch ist es zum beispiel möglich, Online-Ranglisten zu integrieren.

# Integrierung in den JavaEditor

JavaFX wird direkt von dem JavaEditor unterstützt und vor allem auch in den Editor integriert.

# Alternativen

## Java AWT

- https://www.javatpoint.com/java-awt

Java AWT (Abstract Window Toolkit) wurde in den frühen Tagen von Java eingeführt und diente als Grundlage für die GUI-Entwicklung. Allerdings hat es im Vergleich zu JavaFX mehrere Einschränkungen, die es weniger geeignet für die Spieleentwicklung machen.

## LWJGL

- https://www.lwjgl.org/

LWJGL, die “Lightweight Java Game Library”, ist eine Java-Bibliothek, die den plattformübergreifenden Zugriff auf beliebte native APIs ermöglicht, die bei der Entwicklung von Grafik- (OpenGL, Vulkan), Audio- (OpenAL) und Parallel-Computing-Anwendungen (OpenCL) nützlich sind. Dieser Zugriff ist direkt und leistungsstark, aber auch in einer typsicheren und benutzerfreundlichen Schicht verpackt, die für das Java-Ökosystem geeignet ist.

Wenn man also bereits erfahrungen in OpenGL besitzt, sind diese Bindings sehr geeignet und bieten viel Kontrolle über das, was passiert. Wenn man allerdings diese nicht hat, muss man OpenGL erstmal von Grund auf neu lernen.

## LibGDX

- https://libgdx.com/
- Ziehen wir neben JavaFX in Erwägung

LibGDX ist ein vollwertiges Java spiel Entwicklungs Framework, welches auf OpenGL basiert. Es funktioniert *cross-platform* und funktioniert auf Window, Linux, OSX, Android und im Browser. Eine aktive Community und ausführliche Dokumentation sprechen durchaus dafür. Im gegensatz zu JavaFX werden auch "große" und auch kommerzielle Spiele mit LibGDX programmiert.

Wir denken das LibGDX etwas über das hinaus steigt, was wir benötigen. Falls ich irgendwann mal, in einem anderen Rahmen, ein Spiel in Java mal programmieren sollte, werde ich aber auf dieses Framework zurückgreifen.

## Fazit

JavaFX bieten im Vergleich zu den Alternativen eine gute Menge an Funktionen an, aber eben auch nicht zu viel. Ich denke ohne mindestens JavaFX wird es schwierig ein ordentliches Spiel in der Zeit zu programmieren. Da diese Libary nur die grafische Darstellung und Sounds übernimmt, hat es keinen größeren Einfluss darauf, wie wir unser Spiel, von der Logik und dem Aufbau her, programmieren. Auch von der Performance ist JavaFX eine gute Wahl. Es ist bereits seit ungefähr 16 Jahren in der Entwicklung und wurde hinreichend optimiert.

## Quellen


- https://javaeditor.org/doku.php
- https://www.oracle.com/java/technologies/javase-jdk8-doc-downloads.html
- https://fxdocs.github.io/docs/html5/
- https://openjfx.io/javadoc/21/