

JavaFX State of the Technology

JUG
Görlitz 

Wolfgang Weigend

Master Principal Solution Engineer | global Java Team

Java Technology & GraalVM and Architecture

ORACLE Global Services Germany GmbH



Java™

Agenda

- JavaFX Overview
- JavaFX Community
- JavaFX Releases, through 21
- JavaFX 24 Early-Access Builds
- JavaFX Notebook and Rich Text example
- JavaFX Future Releases
- Q&A



JavaFX Overview

- JavaFX is a UI toolkit for developing desktop and mobile applications
 - First delivered in 2011
 - Programming paradigm is scene-graph based with transformations specified at each node
- Multi-Platform
 - Desktop libraries available for Windows, macOS, and Linux
 - Mobile version for iOS and Android using Gluon Mobile
 - Embedded version for linux-arm (e.g., Raspberry Pi) using Gluon Embedded



JavaFX Overview (continued)

Capabilities

- Complete set of UI controls (e.g., button, table, combo-box, menu, dialog, tooltip)
- CSS styling (customize the look of your app)
- Layout containers to position and resize your content (e.g., BorderPane, GridPane)
- Built-in charts
- 2D and 3D graphical shapes, text, images, filter effects
- MediaPlayer (video and audio)
- WebView (browsable web content, Java \leftrightarrow JavaScript bridge, access DOM tree)
- JavaFX/Swing interop: add JavaFX content to a Swing app (or vice versa)
- Properties and bindings API (express relationships between objects)
- Animation can drive updates of any property – even non-graphical ones



JavaFX Overview (continued)

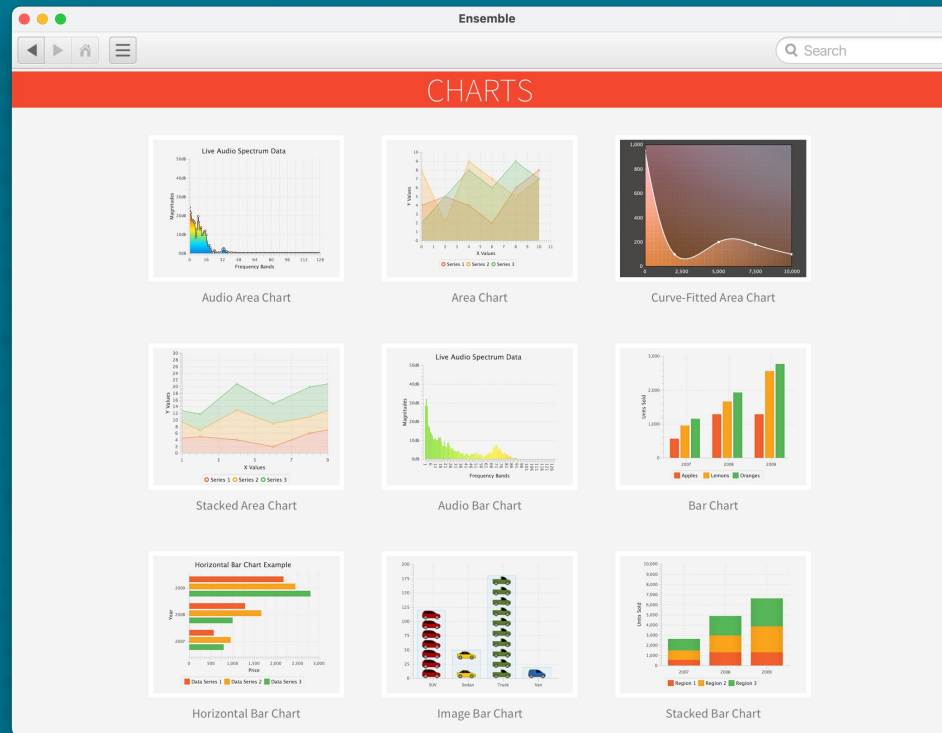
- JavaFX is hardware accelerated under the covers
- Prism Graphics layer takes care of the rendering:
 - OpenGL on Linux
 - OpenGL (for now) on macOS (will move to Metal in the future)
 - Direct3D on Windows
 - Software pipeline as “fall back” if the system doesn’t support acceleration
- Glass window toolkit handles events, window system ops, accessibility
 - Interfaces to platform windowing and input operations



JavaFX Examples

Ensemble

- Ensemble app has examples (with code) of individual features of JavaFX



JavaFX Examples

VirtualThread Visualizer (Project Loom)

Thread Dump Visualizer 2022-1007 - threads2.json

File Tools Help Search: [] X

Container	Owner	Count
<root>		12
ForkJoinPool.commonPool/jdk.internal.vm.SharedThread...		1
ForkJoinPool-1/jdk.internal.vm.SharedThreadContainer@...		8
java.util.concurrent.ThreadPoolExecutor@11326b20/jdk.in...		0
java.util.concurrent.ScheduledThreadPoolExecutor@224e...		0
main/jdk.internal.misc.ThreadFlock@5443d9cd	1	2
xjdk.internal.misc.ThreadFlock@38c0c444	22	3
yjdk.internal.misc.ThreadFlock@57aba509	24	3

Thread Id	Name
32	
33	
27	

```

java.base/jdk.internal.vm.Continuation.yield(Continuation.java:357)
java.base/java.lang.VirtualThread.yieldContinuation(VirtualThread.java:403)
java.base/java.lang.VirtualThread.park(VirtualThread.java:532)
java.base/java.lang.System$2.parkVirtualThread(System.java:2686)
java.base/jdk.internal.misc.VirtualThreads.park(VirtualThreads.java:54)
java.base/java.util.concurrent.locks.LockSupport.park(LockSupport.java:369)
java.base/sun.nio.ch.Poller.poll2(Poller.java:139)
java.base/sun.nio.ch.Poller.poll(Poller.java:182)
java.base/sun.nio.ch.Poller.poll(Poller.java:87)
java.base/sun.nio.ch.NioSocketImpl.park(NioSocketImpl.java:175)
java.base/sun.nio.ch.NioSocketImpl.park(NioSocketImpl.java:196)
java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:304)
java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:348)
java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:789)
java.base/java.net.Socket$SocketInputStream.read(Socket.java:1825)
java.base/java.net.Socket$SocketInputStream.read(Socket.java:1819)
Test2.read(Test2.java:49)
Test2.lambda$x$3(Test2.java:29)
java.base/java.util.concurrent.FutureTask.run(FutureTask.java:317)
java.base/java.lang.VirtualThread.run(VirtualThread.java:287)
java.base/java.lang.VirtualThread$VThreadContinuation.lambda$new$0(VirtualThread.java:174)
java.base/jdk.internal.vm.Continuation.enter0(Continuation.java:327)
java.base/jdk.internal.vm.Continuation.enter(Continuation.java:320)

Owner Thread
id=#32
java.base/jdk.internal.vm.Continuation.yield(Continuation.java:357)
java.base/java.lang.VirtualThread.yieldContinuation(VirtualThread.java:403)
java.base/java.lang.VirtualThread.park(VirtualThread.java:532)
java.base/java.lang.System$2.parkVirtualThread(System.java:2686)
java.base/jdk.internal.misc.VirtualThreads.park(VirtualThreads.java:54)
java.base/java.util.concurrent.locks.LockSupport.park(LockSupport.java:369)
java.base/jdk.internal.misc.ThreadFlock.awaitAll(ThreadFlock.java:315)
jdk.incubator.concurrent/jdk.incubator.concurrent.StructuredTaskScope.implJoin(StructuredTaskScope.java:460)
jdk.incubator.concurrent/jdk.incubator.concurrent.StructuredTaskScope.join(StructuredTaskScope.java:480)
Test2.x(Test2.java:31)
Test2.lambda$main$1(Test2.java:20)
java.base/java.util.concurrent.FutureTask.run(FutureTask.java:317)
java.base/java.lang.VirtualThread.run(VirtualThread.java:287)
java.base/java.lang.VirtualThread$VThreadContinuation.lambda$new$0(VirtualThread.java:174)
java.base/jdk.internal.vm.Continuation.enter0(Continuation.java:327)
java.base/jdk.internal.vm.Continuation.enter(Continuation.java:320)

```



JavaFX Examples

JMail demo (desktop app using JavaFX / Swing)

The screenshot shows a desktop application window titled "Compose: Re: JavaOne 2022 - JMail". The window contains several UI elements that are annotated with labels and arrows:

- Buttons:** Points to the "Send" and "Save" buttons at the top left.
- Label:** Points to the "To:" label in the recipient field.
- TextField:** Points to the "Subject:" label in the subject field.
- HTML Editor:** Points to the rich text editor toolbar, which includes a "Paragraph" dropdown, a font size dropdown set to "12 pt", and buttons for bold, italic, underline, strikethrough, and bulleted list.
- JavaFX Stage:** Points to the top right corner of the window frame.
- Button:** Points to a paperclip icon in the top right corner, representing an attachment button.

The email content area displays the following text:

On 10/4/22 09:01 AM, kcr@openjdk.org wrote:

----- Included message -----

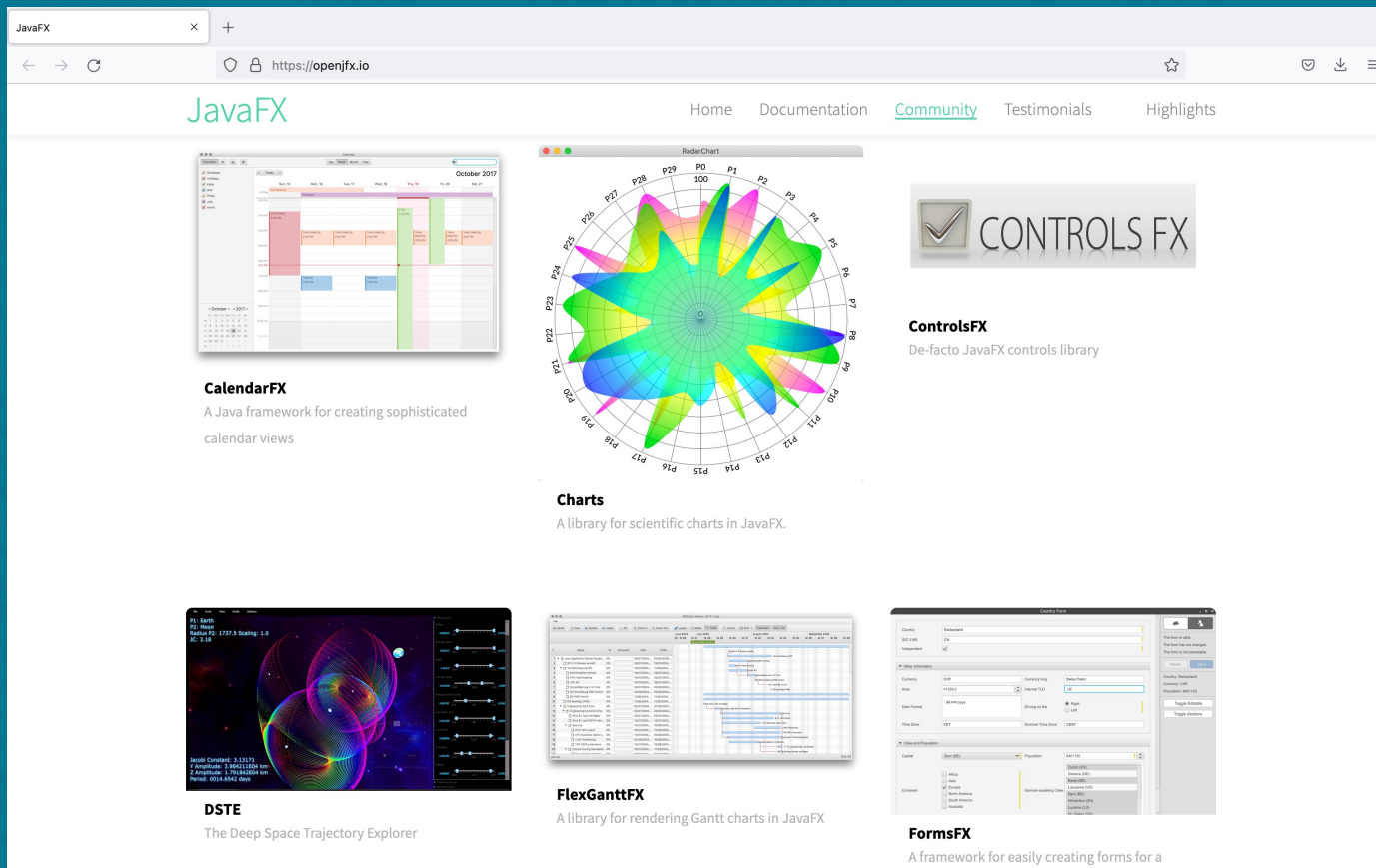
Hi Phil,

And as well as the attachments we have to think about the UI itself.
We can use Swing and FX components where we think each makes sense.
Like the folder list might work well using a JavaFX TreeTableView.
We just have to embed it in a JFXPanel if the parent container is a Swing one.

-Kevin

JavaFX Examples

Additional examples on openjfx.io

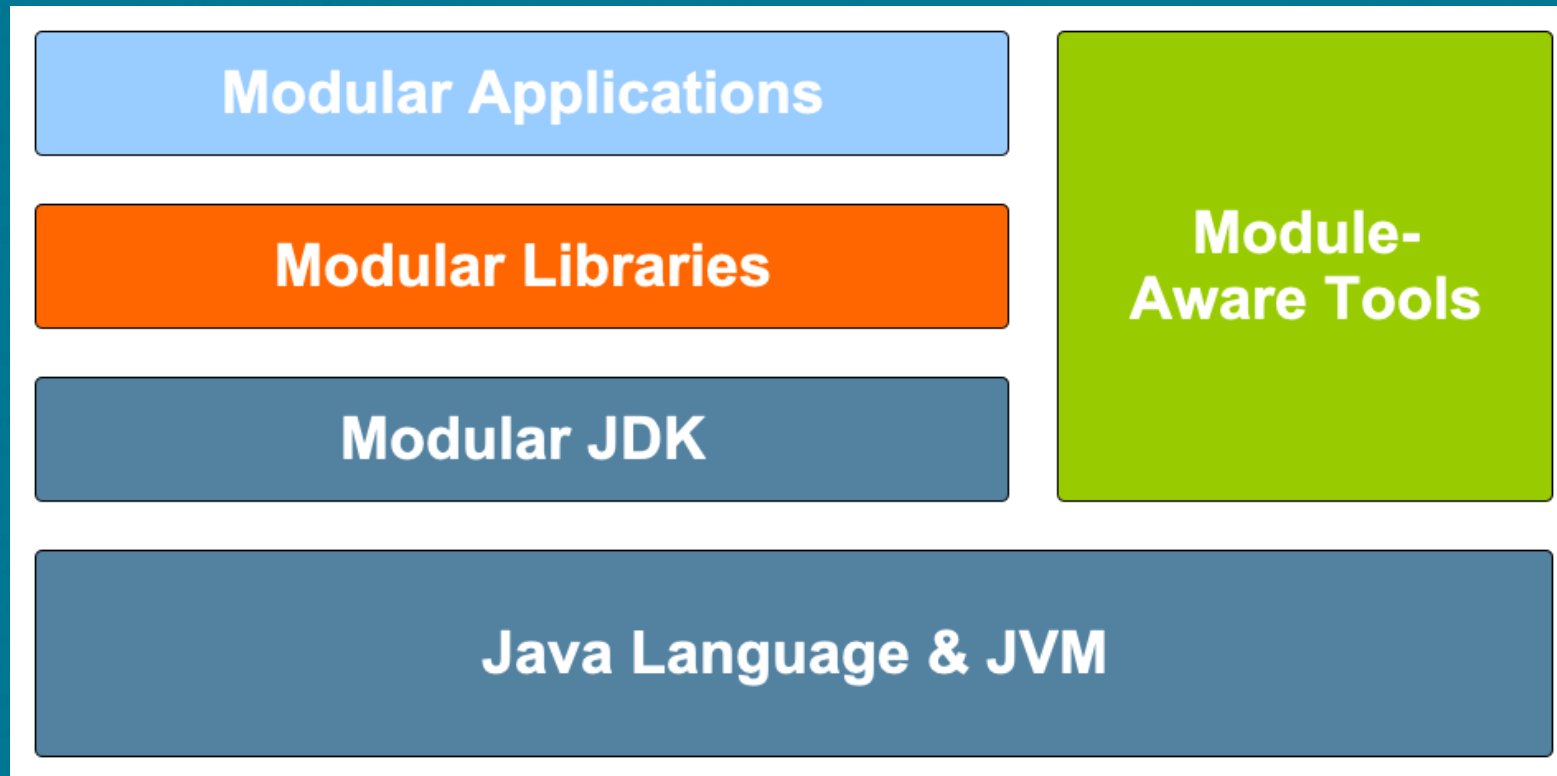


The screenshot shows the openjfx.io website with a navigation menu including Home, Documentation, Community, Testimonials, and Highlights. The main content area features several example cards:

- CalendarFX**: A Java framework for creating sophisticated calendar views. The image shows a calendar for October 2017 with various events and tasks.
- Charts**: A library for scientific charts in JavaFX. The image shows a RadarChart with multiple data series plotted on a circular scale.
- ControlsFX**: De-facto JavaFX controls library. The image shows a logo with a checkmark and the text "CONTROLS FX".
- DSTE**: The Deep Space Trajectory Explorer. The image shows a complex 3D visualization of orbital paths in space.
- FlexGanttFX**: A library for rendering Gantt charts in JavaFX. The image shows a Gantt chart with multiple tasks and their durations.
- FormsFX**: A framework for easily creating forms for a. The image shows a complex form with various input fields, buttons, and a sidebar.



Modular Development with the JDK “after eight”



Building and Running Your JavaFX Program

- JavaFX is available as standalone bundle that runs with a JDK from Oracle or other JDK vendors
- Download the JMODs and use **jlink** to create a custom Java runtime:
 - Includes the javafx modules
 - Optionally includes your app, if modular
- Use **package** to create an application package for distribution



Use `jlink` to include JavaFX in JDK

- Download & unzip `jmods` for your platform from <https://jdk.java.net/javafx21>
- Run `jlink` to produce a JDK that includes the JavaFX modules

```
$ jlink --output jdk-21+javafx-21 \  
      --module-path $JAVA_HOME/jmods:javafx-jmods-21 \  
      --add-modules ALL-MODULE-PATH
```

- Alternatively, `jlink` can produce a custom JDK with only modules you need
 - Use this when you know *exactly* what your application depends on

```
$ jlink --output myjdk21 \  
      --module-path javafx-jmods-21 \  
      --add-modules javafx.controls,java.desktop,java.logging
```



Use **jpackage** to Create a Distributable Application

- A basic **jpackage** example looks like:

```
$ jpackage --main-jar myapp.jar --runtime-image jdk-21+javafx-21 \  
    --name MyApp --input myinputdir --dest myoutdir
```

- This will create MyApp.exe on Windows, MyApp.dmg on macOS, and either MyApp.rpm or MyApp.deb on Linux (depending on distro)
 - Override as needed using appropriate options to create a macOS .pkg, Windows .msi, etc
 - Add platform-specific options to create desktop icons, shortcuts, file associations, etc



Alternative: Use maven (mvn) to build your app

- List JavaFX modules and versions you want in pom.xml like this:

```
<dependencies>
  <dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-controls</artifactId>
    <version>21</version>
  </dependency>
</dependencies>
```

- Run the application using **mvn**
 - Maven downloads javafx modules, including platform natives
 - See <https://openjfx.io/openjfx-docs/#maven> for more information
- You can still use **jpackage** to create a distributable app



JavaFX Community

- JavaFX is developed in the OpenJFX project within OpenJDK
 - Source code available on GitHub: <https://github.com/openjdk/jfx>
 - Mailing list for discussion: openjfx-dev@openjdk.org
- Contributions:
 - We welcome bug fixes from the community
 - We also accept enhancements...with a *much* higher bar. Some examples:
 - Marlin rasterizer
 - Public Robot API
 - VirtualFlow API for 3rd-party controls
 - PointLight attenuation, SpotLight, DirectionalLight
 - Map, FlatMap, and OrElse fluent bindings
 - See: <https://github.com/openjdk/jfx/blob/master/CONTRIBUTING.md>



JavaFX Community – New Features

- New features need significant effort and commitment by contributor:
 - Not simply proposing a change that *your* app would benefit from
 - We don't want “drive-by” contributions
 - Think in terms of API “stewardship”
- Inclusion of new features guided by OpenJFX project leads:
 - Kevin Rushforth (Oracle) and Johan Vos (Gluon)
- Overarching goals for new features:
 - Consistency in the core APIs
 - Maintainability
 - Ability to implement on a wide range of platforms (including mobile devices)



JavaFX Release Model

- Same 6-month cadence as OpenJDK
 - JavaFX 11-20 released along with corresponding JDK release
 - JavaFX 21 released **19th of September 2023** (along with JDK 21)
- As with the JDK, our release model is a “train” model
 - Features go in when ready (they catch the next train)
- Mainline jfx repo is currently open for JavaFX 22 fixes
- **JavaFX N will run on JDK N-1 and later**
 - However, we won't bump the minimum version without good reason
 - JavaFX 11 through 19 all require JDK 11 LTS and later
 - JavaFX 20 require JDK 17 LTS and later
 - JavaFX 21 will run on JDK 17 LTS - but JDK 21 LTS is recommended



JavaFX 17 Highlights

- Released in September 2021
- Included:
 - 11 enhancements
 - 83 bug fixes
- Notable features:
 - 3D SpotLight type
 - Load images from inline data-URIs
 - Load stylesheets from inline data-URIs
 - Print to file
 - Query states of CAPS LOCK and NUM LOCK keys
 - Multiple screen support in window toolkit for embedded platforms
 - [Performance] Add “treeShowing” listener to scene graph only when needed

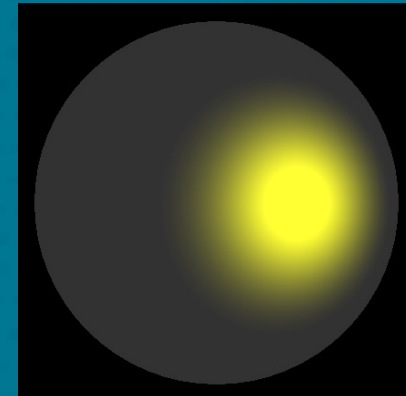


JavaFX 17 Highlights

3D SpotLight type

- Subclass of PointLight
- Adds direction and a cone of influence for more realistic lighting effects

```
var spotLight = new SpotLight(Color.YELLOW);  
spotLight.setTranslateX(100);  
spotLight.setTranslateZ(-200);  
spotLight.setDirection(new Point3D(-100, 0, 200));  
spotLight.setOuterAngle(15);
```



JavaFX 17 Highlights

Load images or stylesheets using data URI

- Encode the data directly in a URI using base64 encoding
- Example loading a PNG file from a data URI:



```
var imageURIString = "data:image/png;base64, iVBORw0KGgoAAAANSUgAAAIAAAACACAIAAABMXPacAAABhWlDQ" +  
    "1BJQ0MgcHJvZmIsZQAACKJF9kT1Iw0AcxV9TpSoVBQuKqGSoThZERRylikWwUNoKrTqYXPoFTRqSFBdHwbXg4Mdi1" +  
    "cHFwVcHV0EQ/ABxdHJSdJES/5cUWsR6cNyPd/ced+8AoVpkqtK2AaiaZcQjYTGvXhV9r+jECPoxjF6JmXo0sZhey" +  
    "/F1Dw9f70I8q/W5P0e3kjEZ4BGJ55huWMQbxDObls55nzjA8pJCFE48btAFiR+5Lrv8xjnnsMAzA0YyPk8cIBZzTS" +  
    "w3McsbKvE0cVBRNcoXUi4rnLc4q8Uyq9+Tv9Cf0VYSXKc5hAiWEEUMImSUUJARFkK0aqSYiNN+uIV/0PHHyCWTqwB" +  
    "GjgWUoEJy/OB/8LtbMzs16Sb5w0D7i21/jAK+XaBwse3vY9uunQDeZ+BKa/hLVWD2k/RKQwseAT3bwMV1Q5P3gMsd" +  
    "YOBj1wzJkbw0hWwWeD+jb0oDfbdA15rbW30fpw9Akrpavge0DoGxHGwvt3h3R3Nv/56p9/cDmw9yt0mXv5kAAAAJc" +  
    "EhZcwAACxMAAAAsTAQCanBgAAAAHdElNRQfmCgEMKB3JhBbsAAAAzUIEQVR42u3cwQkAMAwDsbj77+zOYdBNEBD+Jj" +  
    "de2+n73wkAAAEIAAABACAAAAQAAACAEEAAAAGAAAEIAAABACAAAAQAAACAEEAAAAGAAAEIAAABACAAAAQAAACAEE" +  
    "AAAgAAAEIAAABACAAAAQAAACAEECX9f/7SSxAAAAIAAABACAAAAQAAAEAAAAGBAAAAIAAABACAAAAQAAAEAAA" +  
    "AAQAgAAAEAAAAGBAAAAIAAABACAAAAQAAAEAAAAGBAAAAIAAABACAA632K6Ab8xSXNnAAAAABJRu5ErkJggg==" ;  
var image = new Image(imageURIString);
```



JavaFX 18 Highlights

- Released in March 2022
- Included:
 - 10 enhancements
 - 68 bug fixes
- Notable features:
 - H.265/HEVC media codec
 - 3D DirectionalLight type
 - Transparent backgrounds in WebView
 - CSS styling for Node's "managed" property
 - Factory methods for Border and Background

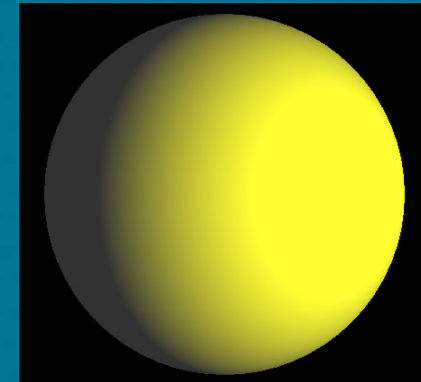


JavaFX 18 Highlights

3D DirectionalLight type

- Used to represent a lights that are very distant (think sunlight)
- Illuminates objects equally in a specific direction

```
var dirLight = new DirectionalLight(Color.YELLOW);  
dirLight.setDirection(new Point3D(-0.5, 0, 0.5));
```



JavaFX 19 Highlights

- Released 13th of September 2022
- Included:
 - 5 enhancements
 - 59 bug fixes
- Notable features:
 - Add H.265/HEVC to HTTP Live Streaming
 - Map, FlatMap, and OrElse fluent bindings
 - Add :focus-visible and :focus-within CSS pseudo-classes
 - [Performance] Optimize observable ArrayList creation in FXCollections



JavaFX 19 Highlights

Map, FlatMap, and OrElse fluent bindings

- Similar in concept to `java.util.Optional`, but for bindings
- More easily express complex binding relationships
- Example:

```
ObservableValue<Boolean> isShowing = listView.sceneProperty()  
    .flatMap(Scene::windowProperty)  
    .flatMap(Window::showingProperty)  
    .orElse(false);
```



JavaFX 19 Highlights

Add :focus-visible and :focus-within CSS pseudo-classes

- :focus-visible used to style (e.g., highlight) a control differently depending on whether the focus should be visible
 - Set when focus is gained via keyboard navigation (and cleared when node loses focus)
 - Not set if focus is gained via mouse or programmatically
- :focus-within is set when a node or any descendant has focus
 - Allows controls with structure to apply a different style when they are “in focus”, even if not directly
- Either of these can be used in a style sheet



JavaFX 20 Highlights

- Released 21st of March 2023
- Included:
 - 8 enhancements
 - 84 bug fixes
- Notable features:
 - Marlin 0.9.4.6 (fixes bugs, improves performance and stability)
 - Constrained resize policies for TableView / TreeTableView
 - Improve the lifecycle of UI controls skins (Skin::install)
 - Simplified, deterministic way to manage listeners (ObservableValue::when)



JavaFX 20 Highlights

Constrained resize policies for TableView / TreeTableView

- Existing constrained resize policy was not flexible nor robust enough
- New policies added to control how columns are resized
 - Better control over how the space is distributed among the other columns when resizing
 - Implementation fixes a number of long-standing bugs
 - Extensible, so an application can define their own policy
- Existing `CONSTRAINED_RESIZE_POLICY` deprecated
 - Maps to the equivalent new policy, but without the bugs



JavaFX 20 Highlights

Simplified, deterministic way to manage listeners

- New “when” method added to ObservableValue:

```
ObservableValue<T> when(ObservableValue<Boolean> condition)
```

- Returned ObservableValue is updated only when condition is true

- Example:

```
var condition = new SimpleBooleanProperty(true);  
var longLivedProperty = new SimpleStringProperty("A");  
var whenProperty = longLivedProperty.when(condition);
```

- whenProperty will “observe” longLivedProperty while condition is true.
- Once condition is false, whenProperty is eligible for GC, if no other strong refs



JavaFX 21 and higher

- General Availability
 - JDK 21 available since 19th of September 2023
 - JavaFX 21 run on JDK 17 and JDK 21 (LTS) is recommended
 - <https://wiki.openjdk.org/display/OpenJFX>
- Mainline GitHub openjdk/jfx repo currently for JavaFX 22 development
 - <https://github.com/openjdk/jfx>
- JavaFX 22 GA on 19th of March 2024, along with JDK 22
- JavaFX 23 GA on 17th of September 2024, along with JDK 23
- **JavaFX 24 GA - Proposed schedule in March 2025, along with JDK 24**
 - Further milestones will be published after the JDK 24 schedule is published



JavaFX 21

Small enhancements for JavaFX 21

- `Toolkit::canStartNestedEventLoop` – DONE
- Improve performance by eliminating unused listeners – DONE
- Emoji support – Partially Done
- Remove terminally deprecated JavaFX GTK 2 library
- Bindings that clean up after themselves
- Reproducible builds with `SOURCE_DATE_EPOCH`
- CSS theming



JavaFX 24 Early Access Builds

- Early access builds of JavaFX are available for download
 - Built from mainline jfx repo on GitHub: <https://github.com/openjdk/jfx>
- Gluon provides builds of JavaFX 24-ea
 - You can download builds from: <https://openjfx.io/>
- Oracle provides JavaFX 24 Early-Access Builds
 - You can download builds from: <https://jdk.java.net/javafx24/>
(the same place you download Early-Access Builds of JDK 24)



JavaFX 24 Early Access by Gluon

Release	GA Date	Latest version	Minimum JDK	Long Term Support	Extended or custom support	Details
24	March 2025	early access	21	no		
23	September 2024	23 (September 2024)	21	no	upon request	details
22	March 2024	22.0.2 (July 2024)	17	no	upon request	details
21	September 2023	21.0.4 (July 2024)	17	yes	upon request	details
20	March 2023	20.0.2 (July 2023)	17	no	upon request	details
19	September 2022	19.0.2.1 (January 2023)	11	no	upon request	details
18	March 2022	18.0.2 (July 2022)	11	no	upon request	details
17	September 2021	17.0.12 (July 2024)	11	until September 2026	upon request	details
16	March 2021	16 (March 2021)	11	no	upon request	details
15	September 2020	15.0.1 (October 2020)	11	no	upon request	details
14	March 2020	14.0.2.1 (July 2020)	11	no	upon request	details
13	September 2019	13.0.2 (January 2020)	11	no	upon request	details
12	March 2019	12.0.2 (July 2019)	11	no	upon request	details
11	September 2018	11.0.20 (July 2023)	11	until September 2023	upon request	details



JavaFX 24 Early Access Builds by Oracle

This is an early access build of the JavaFX 24 Runtime, built from `openjdk/jfx`. It is intended to allow JavaFX application developers to build and test their applications with JavaFX 24 on JDK 24.

The JavaFX runtime is delivered as an SDK and as a set of `jmods` for each platform. You can use the SDK to compile and run JavaFX applications. You can use the `jmods` with `jlink` to create a JDK that includes the JavaFX modules, and optionally, your modular application.

JavaFX 24-ea is designed to work with [JDK 24-ea](#), but it is known to work with JDK 22 and later versions.

We recommend using a recent early access build of [JDK 24](#).

Documentation

- [Browse API Javadoc](#)

Build 9 (2024/9/16)

- [Changes in this build](#)
- [Issues addressed in this build](#)

These early-access builds are provided under the [GNU General Public License, version 2, with the Classpath Exception](#).

	SDK	
Linux/x64	tar.gz (sha256)	58350723 bytes
macOS/AArch64	tar.gz (sha256)	48361132
macOS/x64	tar.gz (sha256)	50982975
Windows/x64	zip (sha256)	48907716
	JMODs	
Linux/x64	tar.gz (sha256)	51445961
macOS/AArch64	tar.gz (sha256)	41648467
macOS/x64	tar.gz (sha256)	44265782
Windows/x64	zip (sha256)	42192528



JavaFX Notebook Demo

- Main idea: **Make it easier for students and others to get started with JavaFX**
- JavaFX enables apps with lots of rich content, but...
 - Simple HelloWorld has a fair bit of ceremony
 - Visualizing data with JavaFX Charts is powerful, but has even more boilerplate
 - There is a learning curve to use many of the more interesting features (e.g., charts, tables)
 - Threading model is challenging for newcomers
- JavaFX Notebook is a proof of concept of some of these ideas
 - Code in JavaFX without initially having to learn the more difficult concepts
 - Easily visualize data using built-in JavaFX charts
- Some of the utilities could eventually go into JavaFX itself



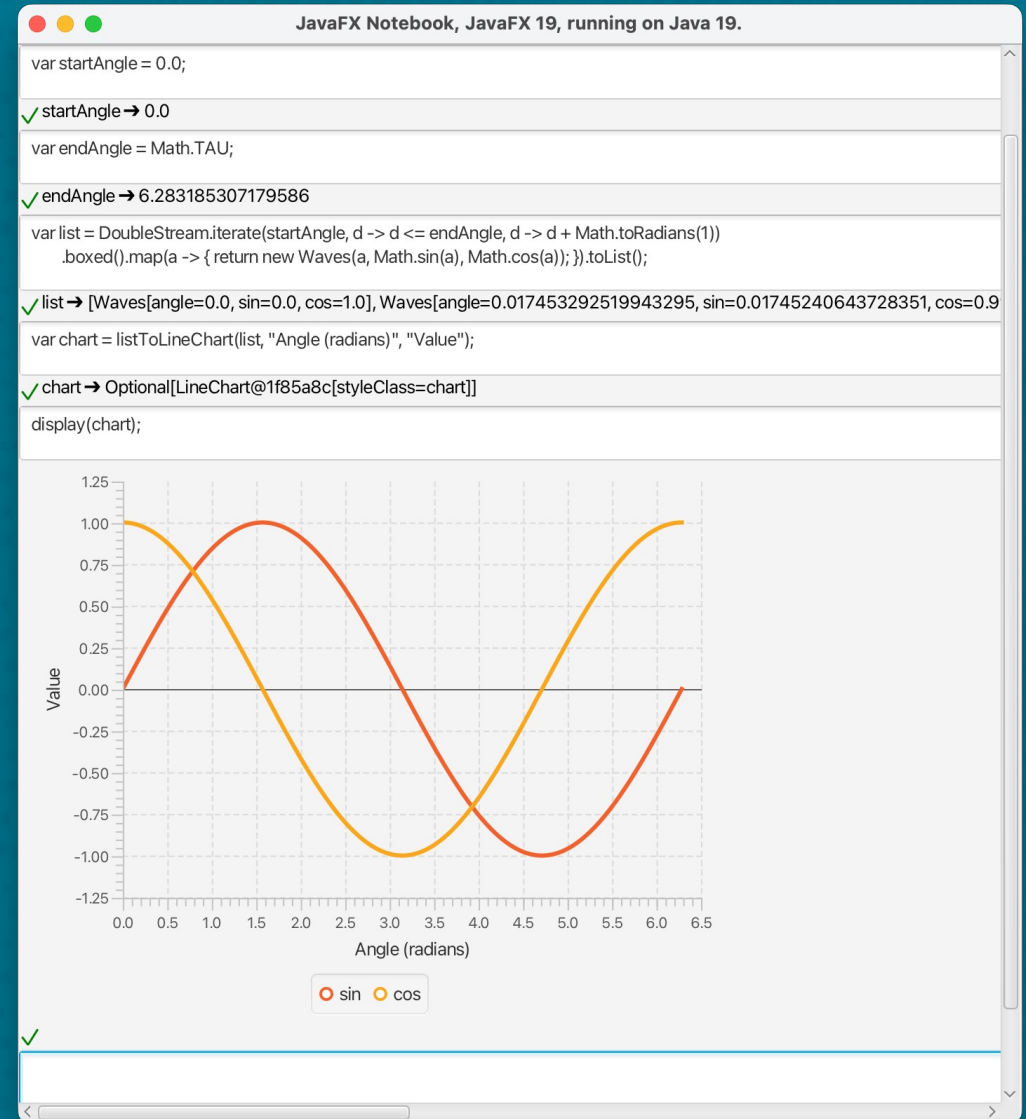
JavaFX Notebook Demo

- Notebook-style application
- Includes utilities for easy creation of tables and charts
- Uses JShell to execute the notebook “cells”
 - No compilation step
- Cells are either code or documentation
 - Java code snippets executed interactively
 - When a code snippet is modified, it and all subsequent snippets re-executed
- ***Enter / edit / delete*** cells
- Persistence (save / load notebooks) using JSON
- Extensibility



JavaFX Notebook Demo

- JavaFX application
 - Runs on JDK that includes JavaFX
- Jshell integration
- Charting utilities
- Uses Jackson to read/write JSON



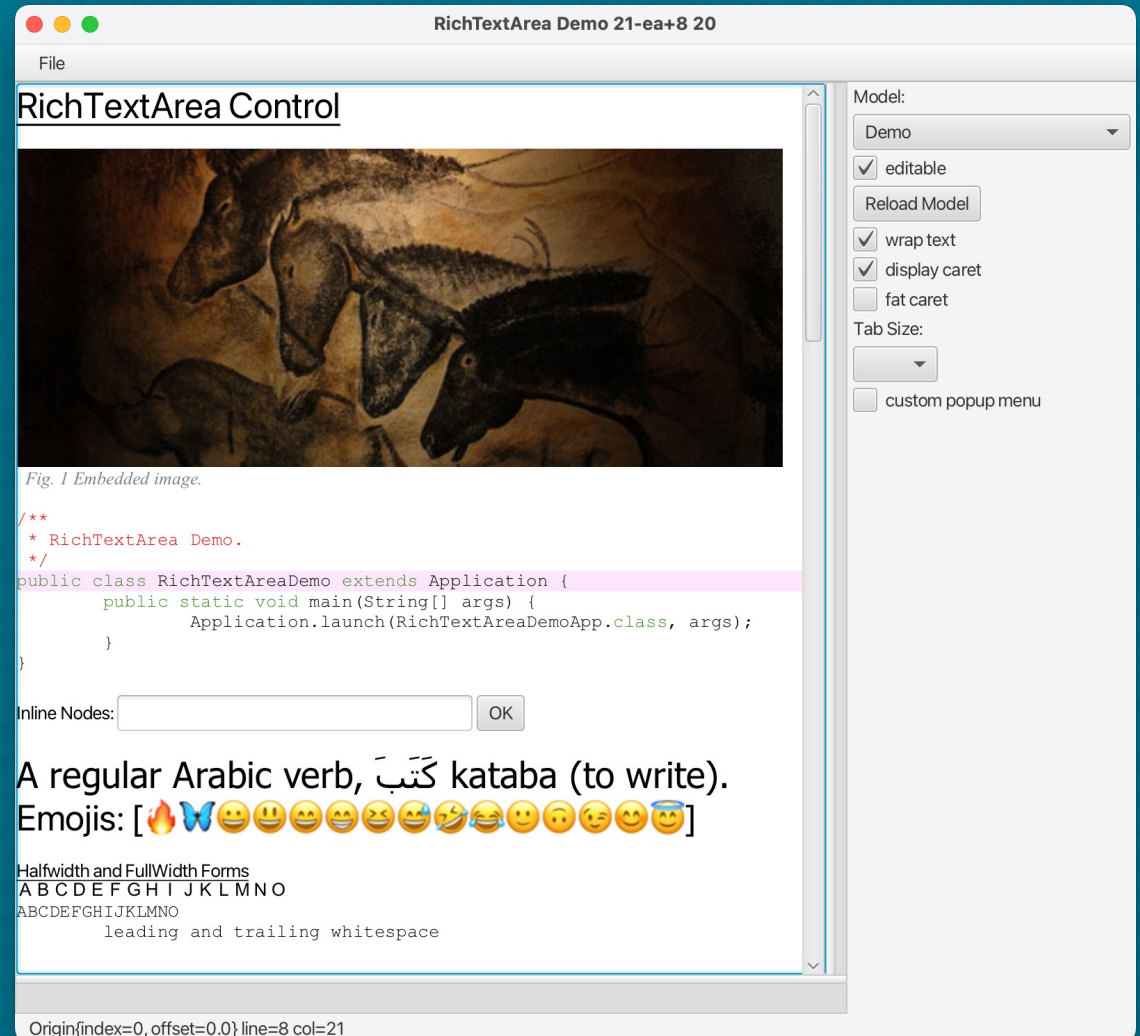
JavaFX Rich Text

- Many applications expect at least some support for stylized text
 - Syntax highlighting for code
 - Inline emphasis, e.g., **color**, **bold**, *italics*, **highlighted**, **monospaced**, **larger**, or smaller.
 - HTMLEditor is too limited (and too heavy-weight)
- We plan to offer a simple, styled RichTextArea control (not full-blown editor)
 - Built in support for basic styling
 - Embedded nodes (e.g., images , panes, tables)
 - Support large models (virtualized)
 - Basic editing
- Extensible
 - We won't provide a full-featured rich text editor, but ...
 - It should be possible to build one on top of what we provide



JavaFX Rich Text Demo

- Prototype RichTextArea control
- Different data models
 - Styled text (not yet editable)
 - Large model (virtualized)
 - Editable plain text (editable)
- Selection & Highlighting
- Navigation & Scrolling
- Copy/paste



Future Releases

General Policies

- Align with JDK releases
 - You can count on JavaFX N working with JDK N-1 and later
 - But we won't bump the minimum JDK "just because"
 - For stability, we intend to use a recent LTS as the minimum JDK
 - For example, JavaFX 19 runs on JDK 11+; JavaFX 20 runs on JDK 17+ (as will JavaFX 21)
 - JavaFX 21 recommended with JDK 21 LTS
- Focus on the core parts of JavaFX
 - Key point: Add functionality when it really makes sense
- Features that can be done by a library on top of FX should be
 - If there is fundamental support missing in core, we can add that support
 - For example, we won't provide a full-blown rich text editor in JavaFX, but will provide a basic RichTextArea along with adding any missing underlying text functionality



Future Releases

Ongoing work

- Fix important bugs – including bug fixes from developer community
- Platform support
 - Updating compilers
 - Fixing critical bugs when new OS versions are released
 - Replacement for deprecated platform APIs (e.g., OpenGL on macOS)
- Updates to WebKit and other third-party libraries
- Small enhancements
- RichTextArea / RichTextField



Future Releases

Platform support

- Continue development of Metal graphics pipeline for macOS
 - Apple has deprecated OpenGL
 - It won't disappear right away, but we need this soon
 - Will leverage the work of Project Lanai
- Replacement for deprecated macOS Accessibility APIs
- DirectX 11/12 support on Windows?

- Native Wayland support on Linux?
 - This will very likely need help from the community
 - Leverage work from Project Wakefield



Future Releases

Possible Ideas

- Ideas from JavaFX Notebook
 - “Quick start” utilities
 - Improvements to Charts
- TableView Improvements
 - column/row freezing
 - column/row spanning
 - commit on focus lost
 - fixed/scrollable columns



Future Releases

Possible Ideas

- Desktop / platform integration
- Multi-resolution image support
- Additional Media features, such as:
 - Closed-captioning (subtitles)
 - Additional audio / video codecs (e.g., Opus, VP9)
 - Support for multiple audio channels
 - Media recording (will take more time)
- What would you like to see?
 - Are you willing to help make it happen?
 - Remember: Features involve more than donating code



JavaFX Releases

➤ Download JavaFX Builds “after eight” here:

- Oracle: <https://jdk.java.net/javafx23/>
- Gluon: <https://gluonhq.com/products/javafx/>



Call to Action

- Download JDK 21 LTS and JavaFX 21
- Download JDK 24-ea and JavaFX 24-ea to test early access builds
- File bug reports at <https://bugreport.java.com/>
- Join the openjfx-dev mailing list



Links

- <https://openjfx.io/> – Community web site
 - Downloads, documentation (javadocs, “getting started” guide), examples
- <https://jdk.java.net/> – Oracle JDK and JavaFX builds
 - Download JDK 21 and JavaFX 21 (and 24 EA)
- <https://github.com/openjdk/jfx> – GitHub repo
 - <https://github.com/openjdk/jfx/blob/master/CONTRIBUTING.md>
- openjfx-dev@openjdk.org – Mailing list to discuss development of JavaFX
 - <https://mail.openjdk.org/mailman/listinfo/openjfx-dev> – Subscribe or view list archives
- <https://openjdk.org/> – OpenJDK home page
- <https://wiki.openjdk.org/display/OpenJFX/Main> – JavaFX Wiki

Danke!

Wolfgang Weigend

Master Principal Solution Engineer | global Java Team

Java Technology & GraalVM and Architecture

ORACLE Global Services Germany GmbH

