
JavaOne™

JavaFX 19 and Beyond

Kevin Rushforth
Java Desktop UI Client Team,
Java Platform Development Group, Oracle
[LRN2615 – JavaOne 2022]

Agenda

- JavaFX Overview
- JavaFX Community
- JavaFX Releases (through 19)
- JavaFX Notebook Demo
- JavaFX 20 ea
- JavaFX Futures

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

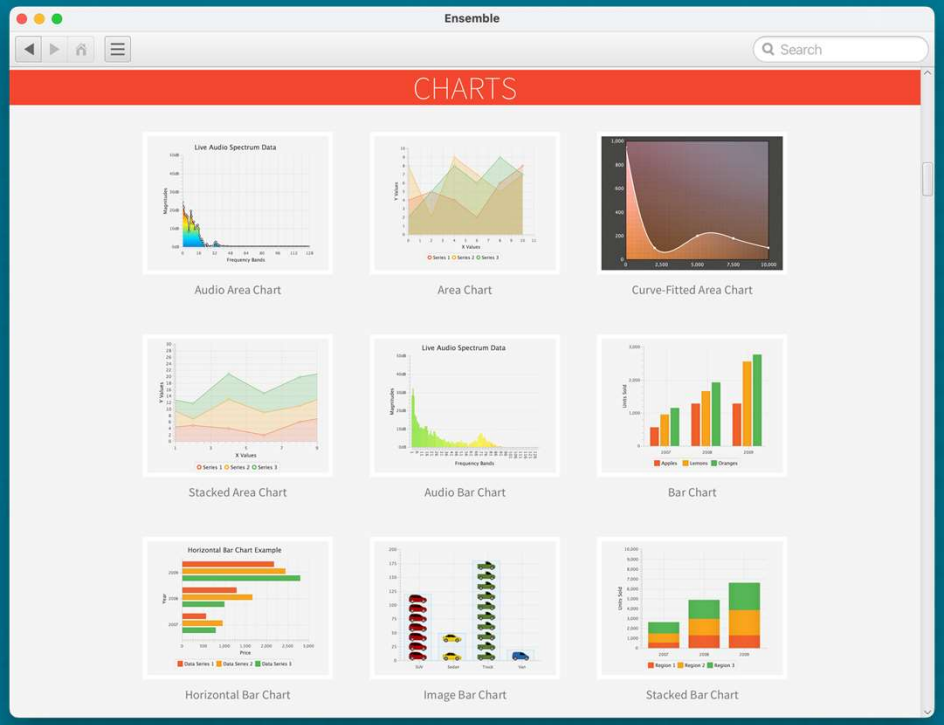
- Complete set of UI controls (e.g., button, table, combo-box, menu, dialog, tooltip)
- CSS styling (customize the look of your app)
- Layout containers to position and resize your content (e.g., BorderPane, GridPane)
- Built-in charts
- 2D and 3D graphical shapes, text, images, filter effects
- MediaView (video and audio)
- WebView (browsable web content, Java <–> 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 HW accelerated under the covers
- Prism Graphics layer takes care of the rendering:
  - OpenGL on Linux
  - OpenGL (for now) on macOS (will move to
  - Direct3D on Windows
  - SW pipeline as “fall back” if the system doesn’t support acceleration
- Glass window toolkit handles events, window system ops, a11y
  - 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

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
x/jdk.internal.misc.ThreadFlock@38c0c444	22	3
y/jdk.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:2606)
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:102)
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:340)
java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:789)
java.base/java.net.Socket$SocketInputStream.read(Socket.java:1025)
java.base/java.net.Socket$SocketInputStream.read(Socket.java:1019)
Test2.read(Test2.java:49)
Test2.lambda$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$2.parkVirtualThread(System.java:2606)
java.base/jdk.internal.misc.VirtualThreads.park(VirtualThreads.java:54)
java.base/java.util.concurrent.locks.LockSupport.park(LockSupport.java:369)
java.base/jdk.internal.misc.ThreadFlock.awaitAll(ThreadFlock.java:315)
jdk.incubator.concurrent/jdk.incubator.concurrent.StructuredTaskScope.implJoin(StructuredTaskScope.java:460)
jdk.incubator.concurrent/jdk.incubator.concurrent.StructuredTaskScope.join(StructuredTaskScope.java:480)
Test2.x(Test2.java:31)
Test2.lambda$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)

Buttons

Label

TextField

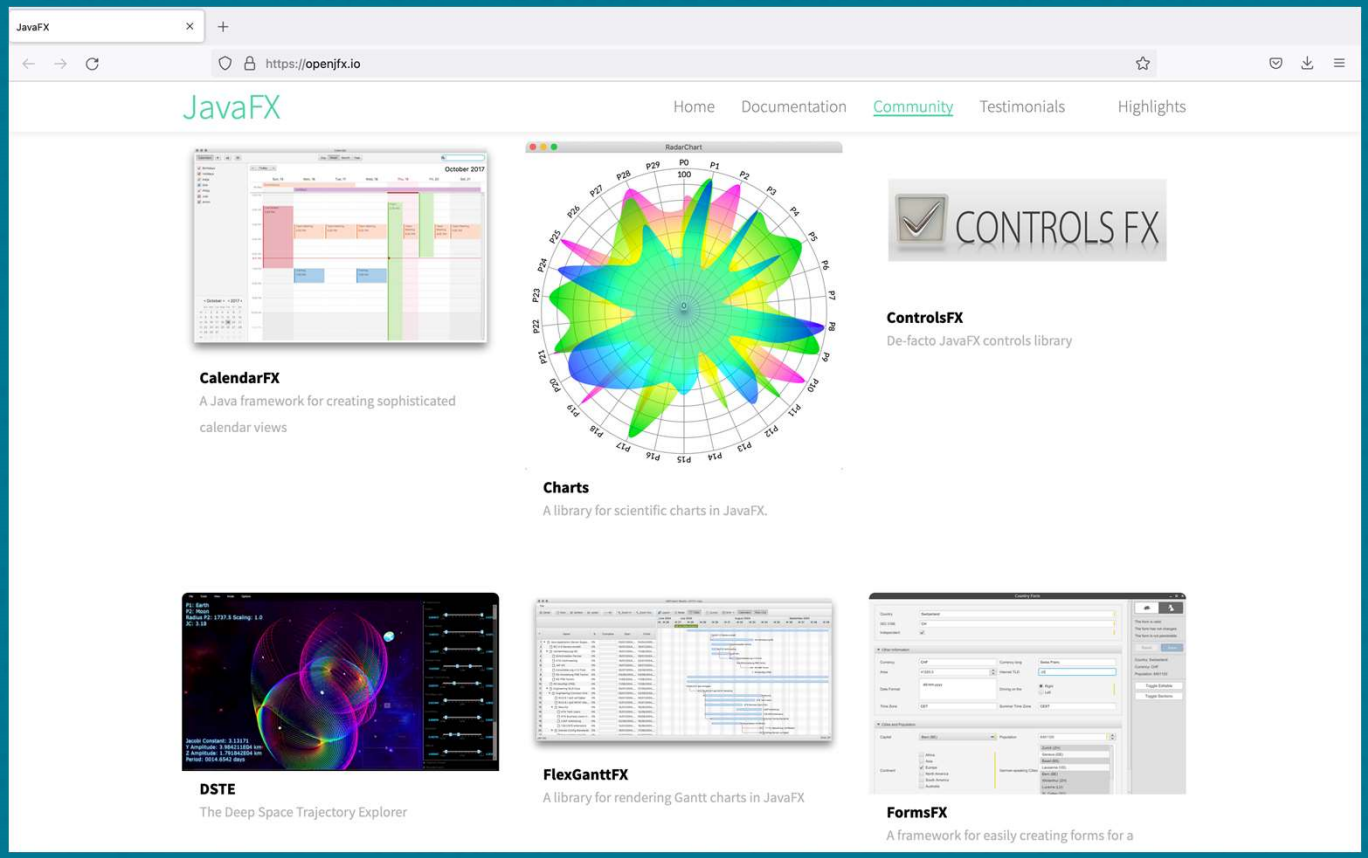
HTMLEditor

JavaFX Stage

Button

JavaFX Examples

Additional examples on openjfx.io



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 jpackage to create an application package for distribution

Use jlink to include JavaFX in JDK

- Download & unzip jmods for your platform from https://openjfx.io/
- Run jlink to produce a JDK that includes the JavaFX modules

```
$ jlink --output jdk-19+javafx-19 \
        --module-path $JAVA_HOME/jmods:javafx-jmods-19 \
        --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 myjdk19 \
    --module-path javafx-jmods-19 \
    --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-19+javafx-19 \
           --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>19</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 11-18 released along with corresponding JDK release
  - JavaFX 19 released in September 2022 (along with JDK 19)
- 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 20 fixes
- OpenJFX N will always run on JDK N-1 and later
  - However, we won’t bump the minimum version without good reason
  - JavaFX 11 through 19 all run with JDK 11 LTS and later
  - JavaFX 20 will run with JDK 17 LTS and later

JavaFX 17 Highlights

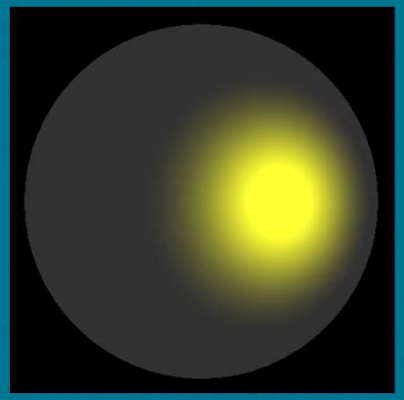
- Released in September 2021
- Included:
  - 11 enhancements
  - 83 bug fixes
- Notable features:
  - 3D SpotLight type
  - Load images from inline data-URIs
  - Load stylesheets from inline data-URIs
  - Print to file
  - Query states of CAPS LOCK and NUM LOCK keys
  - Multiple screen support in window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

JavaFX 17 Highlights

3D SpotLight type

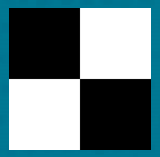
- Subclass of PointLight
- Adds direction and a cone of influence for more realistic lighting effects

```
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 = " iVBORw0KGgoAAAANSUhEUgAAAIAAAACACAIAAABMXPacAAABhWlDQ" +
        "1BJQ0MgcHJvZmlsZQAAKJF9kT1Iw0AcxV9TpSoVBQuKqGSoThZERRylikWwUNoKrTqYXPoFTRqSFBdHwbXg4Mdi1" +
        "cHFWVcHV0EQ/ABxdHJSdJES/5cUWsR6cNyPd/ced+8AoVpkqtk2AaiaZcQjYTGVXhV9r+jECPoxjF6JmXo0sZhEy" +
        "/F1Dw9f70I8q/W5P0e3kjEZ4BGJ55huWMQbxDObls55nzjA8pJCfE48btAFiR+5Lrv8xjnnsMAzA0YyPk8cIBZzTS" +
        "w3McsbKvE0cVBRNcoXUi4rnLc4q8Uyq9+Tv9Cf0VYSXKc5hAiWEEUMImSUUUARFkK0aqSYiNN+uIV/0PHHyCWTqwB" +
        "GjgWUoEJy/OB/8LtbMzs16Sb5w0D7i21/jAK+XaBWse3vY9uunQDeZ+BKa/hLVWD2k/RKQwseAT3bwMV1Q5P3gMsd" +
        "YOBJlwzJkbw0hWWWeD+jb0oDfbdA15rbW30fpw9AkrpavgEODoGxHGWvt3h3R3Nv/56p9/cDmw9yt0mXv5kAAAAJc" +
        "EhZcwAACxMAAAsTAQCanBgAAAAHdElNRQfmCgEMKB3JhBbsAAAAzUlEQVR42u3cwQkAMAwDsbj77+zOYdBNEBD+Jj" +
        "de2+n73wkAAAEAIAAABACAAAAQAAACAEAAAAgAAAEAIAAABACAAAAQAAACAEAAAAgAAAEAIAAABACAAAAQAAACAEA" +
        "AAAgAAAEAIAAABACAAAAQAAACAECX9f/7SSxAAAAIAAABACAAAAQAgAAAEAAAAgBAAAAIAAABACAAAAQAgAAAEAAA" +
        "AAQAgAAAEAAAAgBAAAAIAAABACAAAAQAgAAAEAAAAgBAAAAIAAABACAA632K6Ab8xSXNnAAAAABJRU5ErkJggg==";
var image = new Image(imageURIString);
```


JavaFX 18 Highlights

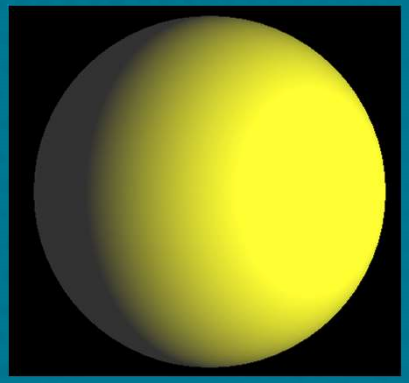
- Released in March 2022
- Included:
  - 10 enhancements
  - 68 bug fixes
- Notable features:
  - H.265/HEVC media codec
  - 3D DirectionalLight type
  - Transparent backgrounds in WebView
  - CSS styling for Node's "managed" property
  - Factory methods for Border and Background

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 September 13, 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);
```

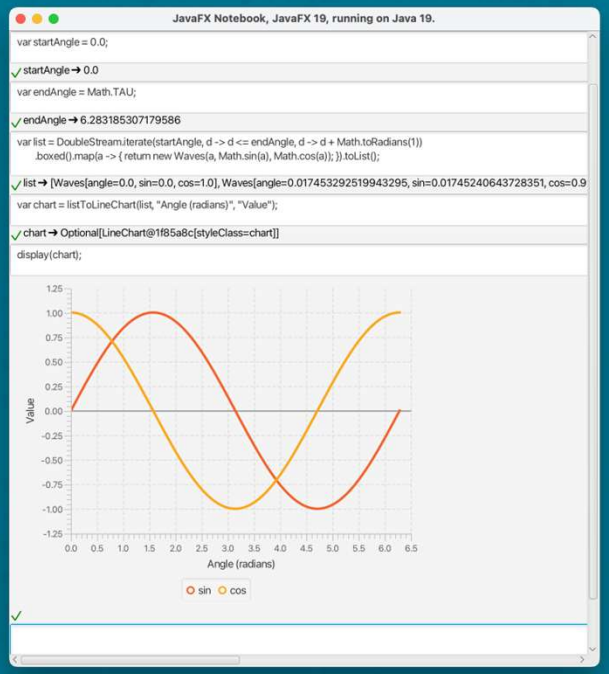

JavaFX 19 Highlights

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 Notebook Demo

- Idea: make it easier 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
- We're exploring make this easier for students and others
- JavaFX Notebook is a proof of concept of some ideas
  - Easy to get started
  - No compilation step
  - Uses JShell to execute the notebook snippets
  - Persistence built into the app



Demo

JavaFX Notebook

JavaFX 20

- Mainline GitHub openjdk/jfx repo currently for JavaFX 20 development
- JavaFX 20 will run on JDK 17 and later (JDK 20 is recommended)
- Proposed schedule:
  - RDP1: Jan 12, 2023 (aka “feature freeze”)
  - RDP2: Feb 2, 2023
  - Freeze: March 2, 2023
  - GA: March 21, 2023

JavaFX 20

Small enhancements underway for JavaFX 20 [1]

- Marlin 0.9.4.6 (fixes bugs, improves performance and stability) – DONE
- Constrained resize policies for TableView / TreeTableView
- Clarify and improve the lifecycle of UI controls skins (Skin::install)
- Improvements to listeners and bindings
  - Simplified, deterministic way to manage listeners
  - Bindings should clean up after themselves
- Reproducible builds with SOURCE_DATE_EPOCH

[1] Enhancements go in when ready; these *might* make 20, but no guarantee

JavaFX 20 Early Access

- Early access builds of JavaFX 20 are available for download
  - Built from mainline jfx repo on GitHub: https://github.com/openjdk/jfx
- Gluon provides builds of JavaFX 20-ea
  - You can download builds from https://openjfx.io/
- Oracle now also provides builds of JavaFX 20-ea
  - You can download builds from https://jdk.java.net/javafx20/ (the same place you download builds of JDK 20)

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 will run on JDK 17+
- 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, a rich text editor can be implemented outside FX, but almost certainly needs some additional text support in the core of FX

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
- Updates to WebKit and other third-party libraries
- Small enhancements

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

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

Future Releases

Possible Ideas

- “Quick start” utilities (taking ideas from JavaFX Notebook)
  - Improvements to Charts
- Text features needed for RichText control
  - Might include styled-text for TextField / TextArea, but *not* a full-blown RichTextEditor
- 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

Call to Action

- Download JDK 20-ea and JavaFX 20-ea
- File bug reports at https://bugreport.java.com/
- Join the openjfx-dev mailing list

Q & A

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 20 and JavaFX 20 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

Thank you

Kevin Rushforth

Java Desktop UI Client Team,
Java Platform Development Group, Oracle


JavaOne™