JavaFX 2.0 Release Notes

Release: JavaFX 2.0

Last Updated: October 2011
Download as PDF

The JavaFX 2.0 release includes the Software Development Kit (SDK), JavaFX Runtime, and JavaFX 2.0 Samples. The SDK provides the tools and technologies for developing JavaFX applications. The runtime enables users to run your JavaFX applications without needing to install the SDK. The samples provide developers a quick overview to the features in JavaFX 2.0.

This release provides Java APIs for JavaFX, which opens the capabilities of JavaFX technology to all Java developers and enables the many existing Java developer tools to be used to create JavaFX applications. In addition, this release provides:

- High performance graphics engine that provides high level support for making rich graphics simple, smooth, and fast.
- Media engine that supports playback of web multimedia content.
- Web component that enables HTML to be embedded in a JavaFX application.
- Refreshed browser plug-in that enables the loading of JavaFX applets based on Prism.
- Extensive set of UI controls, such as Charts, Tables, Menus, and Panes.
- XML-based markup language called FXML for defining user interfaces.
- Sample applications that showcase the features of the JavaFX 2.0 technology.

System Requirements

See the JavaFX System Requirements document for information on the hardware and software prerequisites.

Getting Started

To get started with the JavaFX SDK, review the documentation provided at `http://download.oracle.com/javafx/`. Good starting points include:

- Installing JavaFX - Instructions for installing the JavaFX SDK.
- What is JavaFX? - Introduction to concepts and tools for working with the JavaFX SDK.
- Getting Started with JavaFX - Step-by-step tutorial for creating a JavaFX application.
- API documentation - Output from the Javadoc tool for JavaFX classes.

Known Bugs and Issues

The following sections describe known issues with the code or documentation. Code bugs are arranged by component.

App Model

- **Issue RT-13604:** Standalone application fails to launch when double-clicking if 64-bit JRE is installed.

  Description: Double-clicking on a JavaFX application JAR file fails to launch the application on 64-bit systems with the 64-bit JRE installed when the 64-bit version of the JavaFX Runtime is not installed.

  Workaround: Install the 64-bit version of the JavaFX Runtime.

Control

- **Issue RT-16589:** All em font sizes applied to controls cause double effect.

  Description: Fonts are drawn twice as large as expected when a control and skin have the same CSS style class with a rule that specifies a font size in ems. For example, if you have a base font size of 16px and you have a CSS rule for `Label` and `LabelSkin` with 2em, the font size for `Label` is calculated as 32px and the font size for `LabelSkin` is calculated as 64px. The text for the label is then drawn as 64px.

Workaround: Add the following rule, which is matched by the skin and keeps the font at the size specified for the control:

```
.label > .label { -fx-font-size: 1em; }
```

- **Issue RT-16647:** `getUserAgentStylesheet` is not called in all scenarios.

  Description: The styles for a custom control might not be applied if the styles are defined in a style sheet that is specific to the control.

  Workaround: Load the custom style sheet into the style sheet for the application's scene.

Deployment

- **Issue RT-15819:** File open/save dialog does not show when using `FileOpenService` and `FileSaveService`.

  Description: JNLP API Services (javax.jnlp.*) are not supported in JavaFX 2.0.

  Workaround: In some cases it might be possible to use standard Java APIs, for example, to create or read a file.

Graphics

- **Issue RT-5431:** Mouse events should not be delivered to non-visible components.

  Description: Currently picking does not take into account the front clipping plane of the view frustum in culling picked node. This can result in returning the wrong picked node if a 3D transformed node exists between the viewer and the front clipping plane.

  Workaround: None.
- **Issue RT-14413:** `JFXPanel` does not work in full-screen mode.

  Description: AWT full-screen exclusive mode prevents initialization of the JavaFX D3D pipeline.

  Workaround: Construct a JFXPanel object before entering full-screen mode from any Swing JWindow.
- **Issue RT-15117:** Clipping does not work for objects in 3D

  Description: Objects to which a 3D transform is applied are not clipped correctly. This is a known perspective rendering bug for nVidia and AMD (ATI) GPUs.

  Workaround: None.
- **Issue RT-15181:** Shape with clip set to true does not render correctly with perspective transform and depth test.

  Description: Clipped 3D shapes are not shown correctly when a perspective transformation is performed with depth test set to true.

  Workaround: None.
- **Issue RT-16196:** Dragboard/Clipboard distorts image.

  Description: Some image formats are not copied to the clipboard correctly and may appear distorted.

  Workaround: None.
- **Issue RT-16337:** Animation stops in `JFXPanel` after returning from Windows lock or screensaver.

  Description: On the Windows platform, when the desktop is locked and showing a password prompt, or when a screensaver is active, a `JFXPanel` component is not repainted after returning to the user's desktop.

  Workaround: Resize the JFXPanel component to restart the animation.
- **Issue RT-16397:** Region: asymmetric clipping in an image border that repeats.

  Description: Clipping in an image border is incorrect because the behavior of a `Region` object does not comply with the CSS spec. The `Region` object begins the image slice at the left of the image, however the CSS spec starts the image slice at the center of the image.

  Workaround: None.
- **Issue RT-16398:** Region: distortion in an image border that repeats. Repeated images have the wrong width-to-height ratio.

  Description: When the values defined for the `-fx-border-image-slice` and `-fx-`

border-image-width properties do not match, the image is scaled to fit. However, the image is scaled disproportionately.

Workaround: Size the image so that -fx-border-image-slice and -fx-border-image-width have the same value, so the image does not need to be scaled.

- **Issue RT-16495:** Native Clipboard: passing RTF to a native application does not work.

  Description: An application that tries to put rich text format (RTF) content on the clipboard fails silently and does not leave data formatted as text/rtf on the clipboard.

  Workaround: None.

- **Issue RT-16512:** Region: background position does not work properly

  Description: Setting fx-background-position causes the image to not clip, that is, the image is not included in the calculation.

  Workaround: Manually clip or position the source image.

Media

- **Issue RT-5238:** Negative rate playback is not supported.

  Description: Playing back media in reverse (negative rate) is not yet supported.

  Workaround: None.

- **Issue RT-9100:** FLV and MP3 playback crash Java if the platform does not support the SSE2 instruction set.

  Description: The Java virtual machine could crash if the machine being used to play media does not support the SSE2 instruction set.

  Workaround: Make sure that the machine being used meets the JavaFX system requirements.

- **Issue RT-10611:** Make MediaView resizable.

  Description: The MediaView class cannot be resized.

  Workaround: Create a subclass of MediaView that is resizable and use that class instead.

- **Issue RT-10846:** After stalling, the player does not resume.

  Description: If the media player stalls due to insufficient data such as over a slow network connection, it might not resume playing automatically.

  Workaround: None.

- **Issue RT-13945:** AudioClip doesn`t play short (mp3) audio files.

  Description: Short MP3 clips might not be played by an AudioClip object. The MP3 decoder needs to be primed to be able to produce uncompressed data.

  Workaround: Convert the MP3 clip to uncompressed (PCM) AIFF or WAV format, which are the preferred formats.

- **Issue RT-15044:** Should be able to play media if at least one track is in a supported encoding.

  Description: Media sources might not play if they contain a track that is encoded in a format that is not supported. The media types that are supported are documented as such.

  Workaround: Make sure that all tracks in the media source are encoded in a supported format.

- **Issue RT-16527:** JFXMedia error listener may not catch all error conditions.

  Description: Errors that occur during the initialization of a MediaPlayer object might not be propagated to the interface layer.

  Workaround: None.

- **Issue RT-16675:** Media player state transitions Stalled -> Paused and Stopped -> Paused may not work properly.

  Description: Status transitions from STALLED or STOPPED status to PAUSED status are not always occurring as documented.

  Workaround: None.

- **Issue RT-16809:** Current time is not updated after EOS and seek.

  Description: The internal update of the currentTime property stops if the player is not automatically cycling and the stop time or end of media is reached. After this point, if a manual seek to an earlier time is performed, a listener of currentTime will not receive any more notifications.

  Workaround: Call MediaPlayer.getCurrentTime() directly to get the correct value.

Samples

- **Issue:** A Reference Problem error appears when you open a JavaFX sample project in NetBeans IDE.

  Description: If this is your first time working with JavaFX 2.0 in NetBeans IDE, you see a Reference Problem when you open a Sample project. This error occurs because NetBeans IDE is unable to find the default JavaFX platform.

  Workaround: Create the default JavaFX platform by opening the **File** menu and choosing **New Project**. In the New Project dialog box, choose **JavaFX** in the Categories pane and **JavaFX Application** in the Projects pane, and then click **Next**. When you see Default JavaFX Platform in the JavaFX Platform field, click **Cancel**. The default JavaFX platform is now set for all JavaFX projects.

- **Issue:** NetBeans cannot find the path to JavaFX in a sample project.

  Description: This issue arises under the following conditions:

  - You are **not** running a 32-bit JavaFX SDK with a 32-bit JDK or a 64-bit JavaFX SDK with a 64-bit JDK. For example, your JavaFX is installed in C:\Program Files (x86)\Oracle\JavaFX 2.0 SDK.
  - Your JavaFX SDK is installed in a non-standard location (for example, not on the C drive).
  - You are trying to build the FXML-LoginDemo, and your JavaFX SDK is not installed in C:\Program Files (x86)\Oracle\JavaFX 2.0 SDK.

  Workaround: Open the sample project in NetBeans IDE. In the Projects window, right-click the project name and choose **Properties**. In the Project Properties window, simply click **OK**; no changes are necessary.

  To verify that the path has been set, use an editor to open the project.properties file of your sample JavaFX application. This file is in the nbproject directory. Check that javafx.runtime and javafx.sdk point to where your JavaFX SDK is installed. For example:

  ```
  javafx.runtime=C:\\Program Files\\Oracle\\JavaFX Runtime 2.0
  ```

  ```
  javafx.sdk=C:\\Program Files\\Oracle\\JavaFX 2.0 SDK
  ```

  If javafx.runtime and javafx.sdk still do not point to the right location, update the file manually using the editor.

WebView

- **Issue RT-16536:** NPE when loading web pages that use large textures when rendering.

  Description: If the web page loaded into a `WebView` object makes use of an image or HTML5 canvas whose width or height exceeds a certain limit, the `WebView` object displays a null pointer exception when trying to render the page content.

  The limit is specific to the graphics card used and is usually either 4096 or 8192 pixels. The problem appears only when running the HW pipeline and is not reproducible with the SW pipeline.

  Workaround: None.