WebFX Documentation

Table of Contents
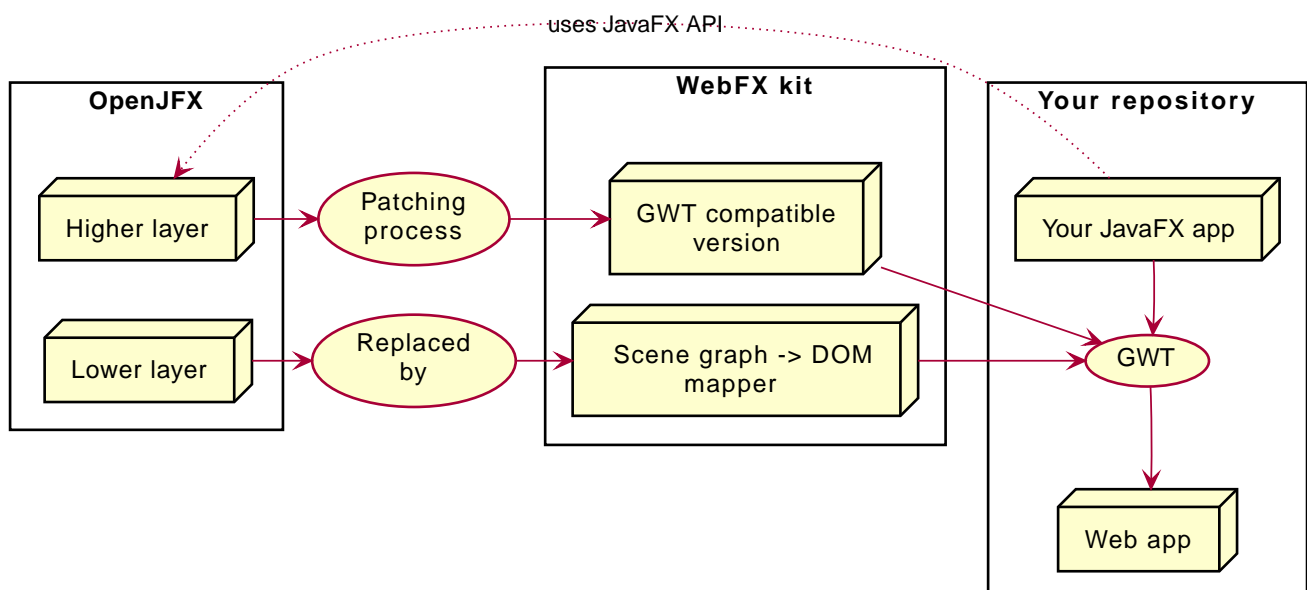
Introduction

What is WebFX?

WebFX is a JavaFX application transpiler powered by GWT. It can transpile a JavaFX application into a traditional self-contained pure JavaScript web app (no plugin or server is required for its execution in the browser).

How it works

The webfx-kit module is the heart of WebFX. It's a modified version of OpenJFX that can be transpiled. This is achieved by patching the higher layer of OpenJFX (which contains the main JavaFX features and API) to make it GWT compatible, and by replacing the lower layer (the graphic rendering pipeline) by a scene graph → DOM mapper (the DOM being finally rendered by the browser).

Limitations

The WebFX kit coverage is for now limited to the essential features of JavaFX. So to successfully compile to the web, your JavaFX app needs to meet these 2 requirements:

- use only the features covered by the WebFX kit (this coverage will increase over the time)
- be compatible with GWT (no reflection, no multi-threading, etc…)

A JavaFX application that meets these 2 requirements is called a *WebFX application*, and it can be transpiled to the web. WebFX achieves this simply by running a GWT compilation of it together with the WebFX kit.

Fast development cycles

You don't need to run a GWT compilation on each development cycle like you would do with a traditional GWT development. Instead, you mainly develop, run, and debug your WebFX application directly in your preferred Java IDE with the OpenJFX runtime (like a standard JavaFX development). You transpile it only from time to time to test the web app.

Cross-platform
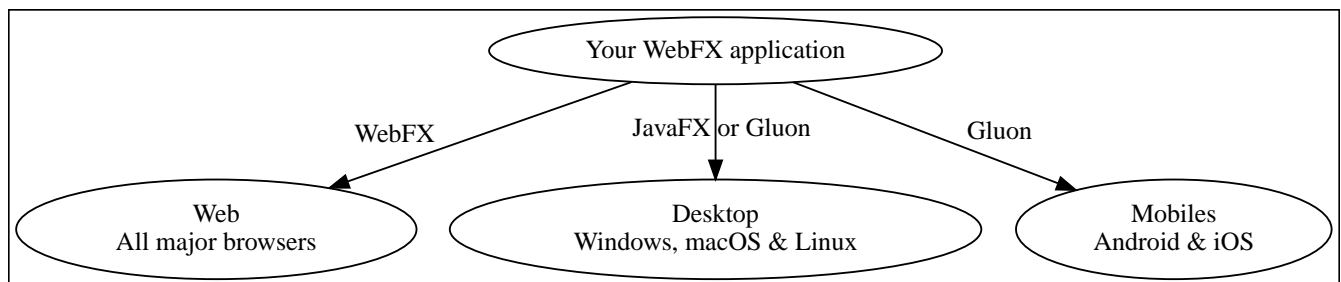
WebFX opens the door to a full cross-platform development from a single source code base. In addition to the web platform, your WebFX application can indeed also be compiled for the desktop & mobiles thanks to the JavaFX & Gluon toolchains.

Getting started

Prerequisite

To develop WebFX applications, you will need the following software already installed on your development machine:

- JDK 13 or above

- Maven

- Git

- Your preferred Java IDE

> **!** Be sure that `java`, `mvn` and `git` are in the execution path of your terminal. The WebFX CLI will invoke these commands without specifying their full path.

Introducing the WebFX CLI

The WebFX CLI is a Command Line Interface for WebFX. It's an essential tool that will assist you developing WebFX applications. It will create your application modules as follows:

```
Your repository
├─ xxx-application (1)
├─ xxx-application-gluon (2)
├─ xxx-application-gwt (3)
└─ xxx-application-openjfx (4)
```

① This module contains the JavaFX code of your application. It is cross-platform (not yet bound to a specific platform) and therefore not executable.

② This module targets the native desktop & mobile platforms. It binds your application with the OpenJFX runtime, and can call the Gluon toolchain to produce the Windows, macOS, Linux, Android & iOS native executables (depending on your OS).

③ This module targets the web platform. It binds your application with the WebFX kit and can call GWT to produce the web app.

④ This module targets the standard desktop platform. It binds your application with the OpenJFX runtime. It is directly executable in your IDE. It can also call the standard JavaFX toolchain to produce the desktop executables (Windows, macOS or Linux) with an embed JVM.

You can create several WebFX applications in the same repository. If your application code grows, you can split your code into more modules. The WebFX CLI will help you to create and maintain all your modules. For each module, it will create and maintain your build files as follows (when applicable to the module):
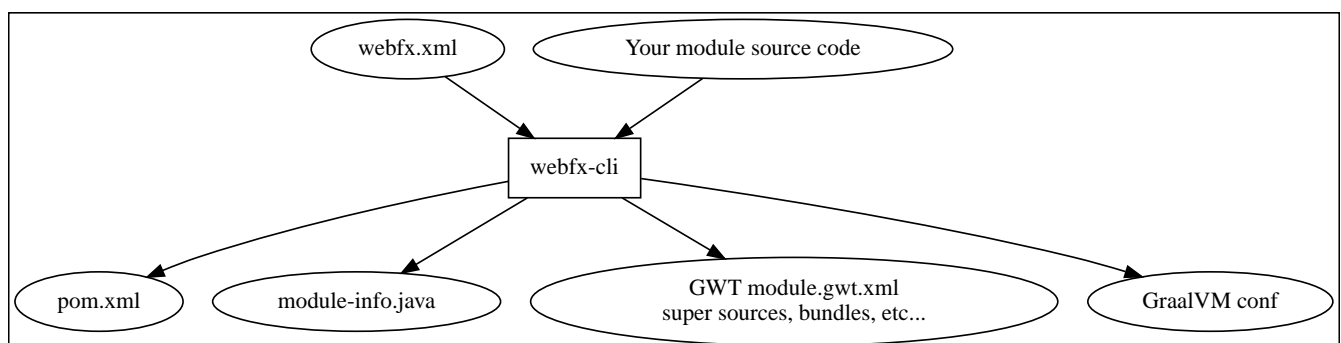
Your inputs will be centralized in the WebFX module files named webfx.xml (same location as

pom.xml), and the WebFX CLI will generate the rest of the build chain from these webfx.xml files. For example, a typical directive in webfx.xml will be:

```
<dependencies>
    <used-by-source-modules/>
</dependencies>
```

This directive is asking the WebFX CLI to generate the list of your dependencies from an analysis of your source code. It will therefore automatically populate the dependencies in pom.xml.

In that process, the WebFX CLI takes over the aspects of cross-platform development: when a feature is platform-dependent (a different implementation exists for different platforms), the tool will pick up the right modules in the final executable modules (the ones matching the platform targeted by the module).

Installing the WebFX CLI

Since we haven't published any release at this stage yet, the way to install the WebFX CLI for now is to clone the webfx-cli repository, and build it with Maven.

> ℹ️ We will distribute the WebFX CLI in a better way on the first official release.

Cloning the webfx-cli repository

HTTPS

```
git clone https://github.com/webfx-project/webfx-cli.git
```

SSH

```
git clone git@github.com:webfx-project/webfx-cli.git
```

Building webfx-cli with Maven

This is achieved by running the Maven *package* goal under the webfx-cli directory:

```
cd webfx-cli
mvn package
```

> ❗ As previously mentioned, WebFX CLI requires JDK 13 or above to successfully compile.

This generates an executable fat jar in the target folder that we can execute with java:

```
java -jar target/webfx-cli-0.1.0-SNAPSHOT-fat.jar --help
```

Creating a permanent *webfx* alias

To easily invoke the WebFX CLI from a terminal, we need to create a permanent *webfx* alias to it. This is done with the following command (to run under the webfx-cli directory):

Linux

```
echo "alias webfx='java -jar $(cd "$(dirname "$1")" && pwd -P)/$(basename
"$1")/target/webfx-cli-0.1.0-SNAPSHOT-fat.jar'" >> ~/.bashrc ①

source ~/.bashrc ②
```

① Adding the alias to the shell profile

② Applying it to the current session

macOS >= Catalina

```
echo "alias webfx='java -jar $(cd "$(dirname "$1")" && pwd -P)/$(basename
"$1")/target/webfx-cli-0.1.0-SNAPSHOT-fat.jar'" >> ~/.zshrc ①

source ~/.zshrc ②
```

① Adding the alias to the shell profile

② Applying it to the current session

macOS < Catalina

```
echo "alias webfx='java -jar $(cd "$(dirname "$1")" && pwd -P)/$(basename
"$1")/target/webfx-cli-0.1.0-SNAPSHOT-fat.jar'" >> ~/.bash_profile ①

source ~/.bash_profile ②
```

① Adding the alias to the shell profile

② Applying it to the current session

Windows PowerShell

```
If (!(Test-Path $profile)) { New-Item -Path $profile -Force } ①

"`r`nfunction webfx([String[]] [Parameter(ValueFromRemainingArguments)] `$params) {
java -jar $((Get-Item .).fullName)\target\webfx-cli-0.1.0-SNAPSHOT-fat.jar `$params
}`r`n" >> $profile ②

If ($(Get-ExecutionPolicy) -eq "Restricted") { Start-Process powershell -Verb runAs
"Set-ExecutionPolicy -ExecutionPolicy RemoteSigned" -Wait } ③

. $profile ④
```

① Creating a PowerShell profile if it doesn't exist

② Adding the alias (implemented as a function) to it

③ Lowering the execution policy if necessary to execute the profile

④ Applying it to the current session

Then you should be able to invoke the CLI tool from the terminal:

```
webfx --help
```

Updating the WebFX CLI to the latest version

You can check for update at anytime by running:

```
webfx bump cli
```

If a new version is available, it will download it and build it.

Creating your first WebFX app

Creating and initializing your repository

Let's create our first WebFX application. We need to create the repository folder and ask the WebFX CLI to initialize it, passing it the groupId, artifactId and version of our application.

```
mkdir webfx-example
cd webfx-example
webfx init org.example webfx-example 1.0.0-SNAPSHOT
```

Creating your application modules

```
webfx create application --class org.example.webfxexample.WebFxExampleApplication
--helloWorld
```

```
webfx-example
├ webfx-example-application
├ webfx-example-application-gluon
├ webfx-example-application-gwt
└ webfx-example-application-openjfx
```

Building your application

```
webfx build
```

Running your application

You can run the OpenJFX version of your application with the following command:

```
webfx -m webfx-example-application-openjfx run
```

You can run the GWT version of your application with the following command:

```
webfx -m webfx-example-application-gwt run
```

Developing in your IDE

We will give the instructions for IntelliJ IDEA, but you should be able to easily transpose them to other Java IDEs such as NetBeans or Eclipse.

Opening the project

Configuring the OpenJFX application

Building and running the GWT application

Making changes

```
webfx update
```