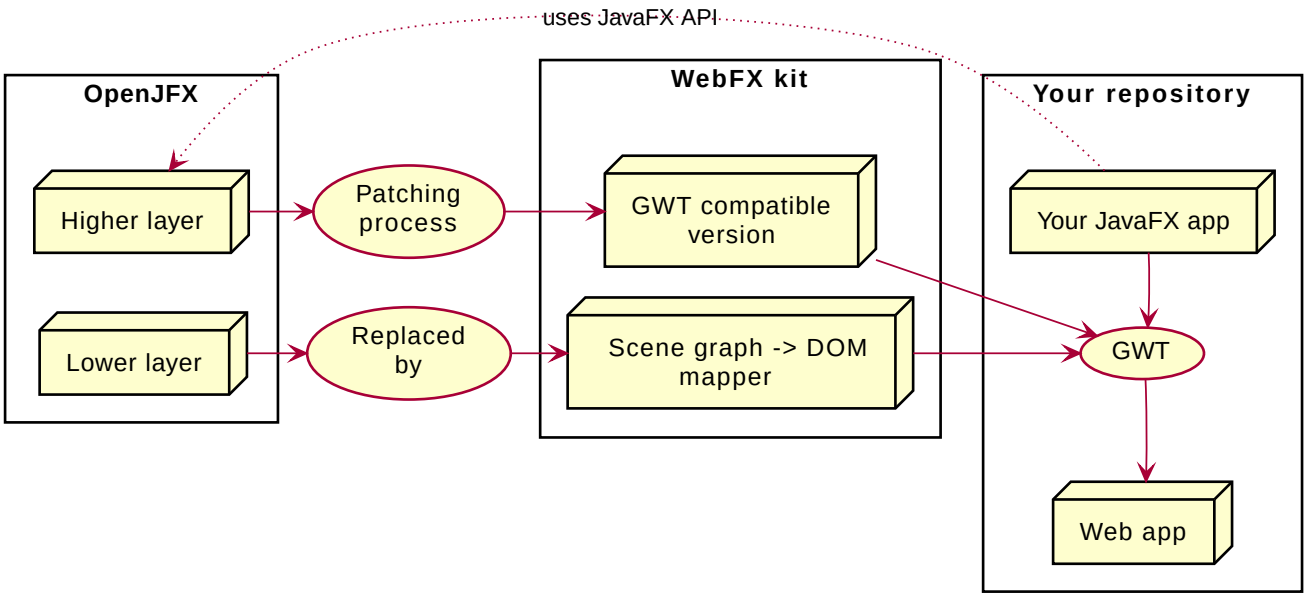
WebFX Documentation

Table of Contents

Introduction

WebFX is a JavaFX application transpiler powered by GWT that produces a pure JS web app (no plugin, no server). It works as follows:



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).

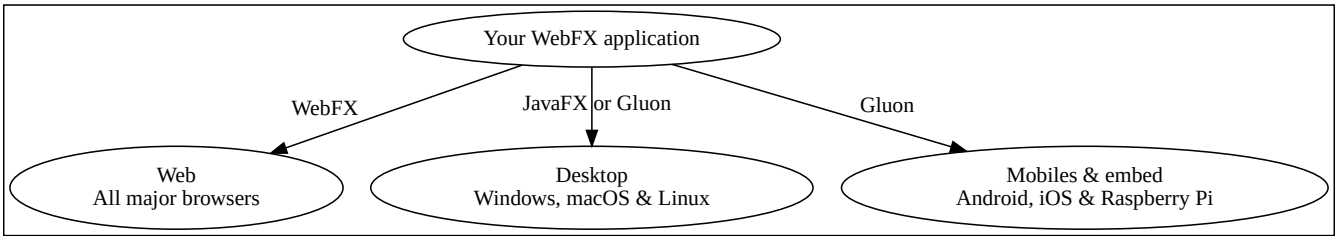
The WebFX kit coverage is for now limited to the essential features of JavaFX. So to successfully compile to the web, your JavaFX app needs to meet these 2 requirements:

- 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. To transpile it, WebFX simply asks GWT to compile it together with the WebFX kit.

You don't need to run a GWT compilation on each development cycle, you can simply develop, run and debug your WebFX application as usual in your IDE with the OpenJFX runtime.

WebFX allows you to do a full cross-platform development from a single source code base. In addition to the web platform, your WebFX application can indeed be compiled also for the desktop, mobiles & embed 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 you can invoke `java`, `mvn` and `git` directly from the terminal. The WebFX CLI will invoke `mvn` and `java` without specifying their full path.

Introducing the WebFX CLI

The WebFX CLI is the Command Line Interface for WebFX. It's an essential tool to 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 not directly executable as it is not

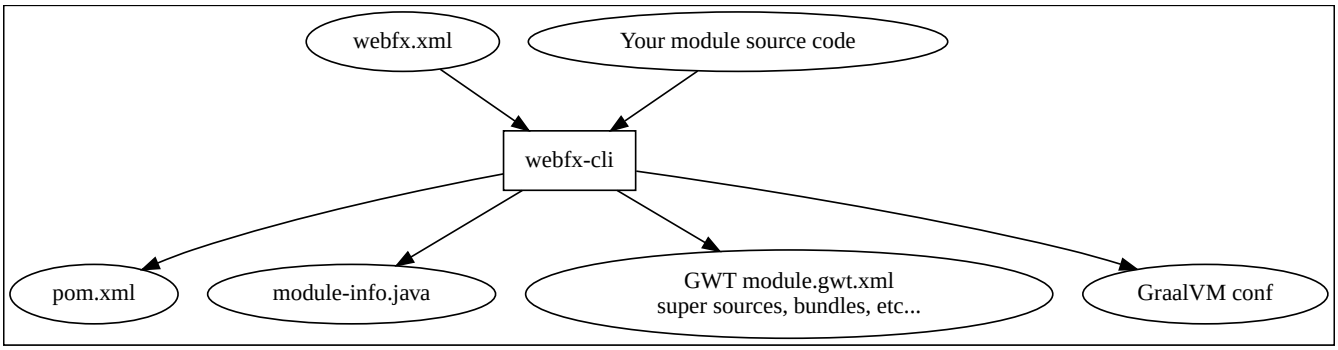
yet bound to a specific platform.

② This module targets the mobile platforms. It will call the Gluon toolchain to produce the Android / iOS executables.

③ This module targets the web platform. It will call GWT to produce the HTML and JS code.

④ This module targets the desktop platform. It will create an executable Java artifact with the OpenJFX runtime. You can also use this module to directly run and debug your application in your IDE.

You can create several WebFX applications in the same repository, such as a front office and a back office for example. Of course, 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):



Your inputs will be centralized in the WebFX module files named webfx.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 automatically generate the list of your dependencies from an analysis of your source code.

The WebFX CLI takes over the aspects of cross-platform development: when a feature is platform-dependent (a different implementations exists for each platform), 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

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

We will distribute the WebFX CLI in a better way later, when we will publish 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 (you will get a build error with lower versions).

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

If later you want to update the WebFX CLI to the latest SNAPSHOT version, you just need to update your local repository and rebuild it with Maven. This is done through the following commands (under your webfx-cli local folder):

```
git pull
mvn package
```

Creating your first WebFX app

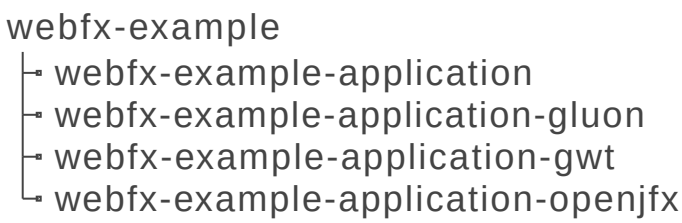
Creating and initializing your repository

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
```



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
```