JavaFX Essentials

Create amazing Java GUI applications with this hands-on, fast-paced guide

Mohamed Taman
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Started with JavaFX 8'
- A synopsis of the book’s content
- More information on JavaFX Essentials
About the Author

Mohamed Taman, chief of architects and software development manager at e-finance, lives in Cairo, Egypt. He graduated in electrical engineering from Faculty of Engineering, Cairo University. He is an experienced Java developer who has worked on the Web, mobile, and IoT for industries, including finance, banking, tourism, government, and healthcare. Before that, he worked with Pfizer, Intercom Enterprise (a Gold IBM partner), Silicon Expert, and Oracle using varied technologies, such as user-facing GUI frontends, backends, mid-tiers, and integrations of large-scale systems.

He enjoys speaking at many conferences evangelizing Java standards and his experience worldwide, as he is a strong Java community member and a Java Champion since 2015.

In addition, Mohamed is a member of Adopts Java EE 8, OpenJDK, and JavaFX programs. He was an executive member of Java Community Process Organisation, being the first African to join its board.

He is also a member of the expert group JSR 354, 363, and 373. He is also the leader of EUGJUG and MoroccoJUG member, a board member of Oracle Egypt Architects Club. He won the 2014 Duke’s choice and 11th annual JCP adopt 2013 awards.

You can read more about the author at http://about.me/mohamedtaman.
Preface

This book, as its title (JavaFX 8 Essentials) suggests, is a pragmatic book that provides you with a robust set of essential skills that will guide you to become confident enough to rapidly build high-performance JavaFX 8 client applications. These applications take advantage of modern GPUs through hardware-accelerated graphics while delivering a compelling, complex, and fancy rich-client GUI for your customer, which will impress them quite a bit.

Learning the JavaFX 8 essentials is the first step to plunging into creating applications that most importantly run on any platform, from the desktop, Web, mobile, tablets, to embedded devices such as Arduino, Raspberry Pi, and multi-core development. Following Java's Write once, run anywhere paradigm, JavaFX also preserves the same. Because JavaFX 8 is written totally from scratch in the Java language, you will feel at home.

Most of the chapters are a fast-paced guide that will help you get a head start on Java GUI programming, leveraging JavaFX 8 and deploying and running on any platform.

While working through the book examples, you will find code is written with JavaFX 8 on Java 8 (yes, Java SE 8) so that the new APIs and language enhancements will help you become a more productive developer. Having said this, it will be handy (and I encourage you to go for this) to explore all of the new Java 8 capabilities.

Finally, yet importantly, you will be able to develop amazing touch-less interactive motion applications with JavaFX that interact with Leap motion devices.
What this book covers

Chapter 1, Getting Started with JavaFX 8, is an introduction to JavaFX 8. It discusses JavaFX 8 as a technology, why you should care about it, its history, core features, and where it can be used.

So it is time to get ready with the right tools and go through the necessary steps to install JavaFX 8 and its supporting development tools. Learn about additional tools that will increase reader productivity in this chapter. As a final verification that we are on the right track, we are going to close the chapter with a simple Hello JavaFX application.

Chapter 2, JavaFX 8 Essentials and Creating a Custom UI, discusses how there is nothing more frustrating than receiving complicated advice as a solution to a problem. Because of this, I have always made it a point to focus on the essentials. In order to render graphics on the JavaFX scene, you will need a basic application, scene, canvas, shapes, text, controls, and colors.

Also, you will learn about JavaFX 8 essential application structures that serve as a backbone to any future application. And finally, we will also explore some Java SE 8 features (such as Lambda, Streams, JavaFX Properties, and so on) to help increase code readability, quality, and productivity.

After getting hands-on experience in creating a structured JavaFX 8 application, wouldn’t it be nice if you could change the UI of your application without altering its functionality? In this chapter, you will learn about theming and how to customize applications by applying various themes (look and feel) and the fundamentals of JavaFX CSS styling.

You will use Scene Builder to create and define UI screens graphically and save them as a JavaFX FXML-formatted file. Finally, you will learn about creating custom controls.

Chapter 3, Developing a JavaFX Desktop and Web Application, covers on how to develop a compelling desktop and Web application that takes advantage of multi-core hardware accelerated GPUs to deliver a high performance UI-based application with an amazing appearance.

As JavaFX is totally written from the ground up in Java, some Java SE 8 built-in core libraries will be used to power our application. Also, you will learn how to package your application as a standalone application to be launched and distributed.

In addition, we will cover the essential core web APIs in any web application levered by JavaFX 8, such as javafx.scene.web.WebEngine and javafx.scene.web.WebView.
We will also discuss the relationship between JavaFX and HTML5, which is important because they complement each other. JavaFX's rich client APIs, coupled with HTML5's rich web content, create a user experience resembling a RIA Web application with the characteristics of native desktop software.

Chapter 4, Developing a JavaFX Application for Android, as we see a rise in non-pc clients, mobile phones and tablets are gaining market share. JavaFX 8 can deliver a rich client application for Web and desktop. If you write a JavaFX application, make sure you want it to run on as many devices as possible. This chapter will give you essential hands-on experience and knowledge about SDKs that allow users to create native applications for Android mobile phones.

Chapter 5, Developing a JavaFX Application for iOS, is an extension to the previous chapter. If you write a JavaFX application for Android, be sure you want it to run on as many iOS devices as possible. This chapter will give you essential hands-on experience and knowledge about SDKs that allow them to create native applications for Apple iOS.

Chapter 6, Running JavaFX Applications on the Raspberry Pi, will provide you with all the necessary skills and knowledge to develop a JavaFX 8 application that runs on a credit card-sized computer, the Raspberry Pi board. As the Internet of things (IoT) has become a hot topic of late. Java was made for the Internet of things literally.

Chapter 7, Monitoring and Controlling Arduino with JavaFX, covers another kind of Internet of everything (IoT). Arduino is an open-source electronics prototyping platform, delivering low-cost prototyping platforms to support both the do-it-yourself concept and the maker movement.

This chapter will provide you with all the necessary skills and knowledge to quickly use JavaFX along with an Arduino board to develop desktop applications for monitoring data coming from the real world or controlling real devices.

Chapter 8, Interactive Leap Motion Apps with JavaFX, will make you learn about gesture recognition. You will discover an awesome gadget, the Leap Motion device, which will allow a touch-less approach to develop enhanced JavaFX applications.

Machine user input interfaces are becoming increasingly less mouse-centric, in favor of multi-touch and even touch-less input. Gestures are one of the ways humans can communicate with machines naturally these days.
Appendix, *Become a JavaFX Guru*, will make you find many useful links and references that will help you gain further knowledge about all things JavaFX.

At the end of this chapter, make sure to check out the many frameworks, libraries, and projects that use JavaFX in production today.
Getting Started with JavaFX 8

JavaFX is Java's next-generation **Graphical User Interface (GUI)** toolkit. It's a platform that makes it easy to rapidly build high-performance Java client-side applications.

JavaFX's underlying engines take advantage of modern GPUs through hardware-accelerated graphics, while providing well-designed programming interfaces, thus enabling developers to combine graphics, animation, and UI controls.

These capabilities allow you to deliver a compelling, complex, and fully customizable client GUI for your customer that will make them quite impressed.

While the original targets of Java were the *embedded* and *client* worlds, since 2006, many reasons pushed the Java language to become the top development platform for the Enterprise world.

But recently, with the JavaFX platform's entrance as the standard client GUI, those original targets have started to gain popularity again.

Although it is much more than just a GUI toolkit, JavaFX allows Java developers to create client applications with compelling user interfaces that easily connect to backend systems.

In addition, JavaFX's flexible FXML support allows you to build **MVC (Model-View-Controller)** architectural pattern applications easily, and use the WYSIWYG approach using the Scene Builder tool.

JavaFX's bindings feature simplified communication between entities and support MVC even further. In addition to that, JavaFX provides fast, customizable UI modeling using CSS.

By adding a full-fledged WebView component with a document model, mapping to Java code is easy and provides great support for 3D and media capabilities.
In this chapter, we are going to cover the following topics:

- What is JavaFX and what are its targeted platforms?
- A walk through JavaFX history
- JavaFX goals, features, and what's new in JavaFX 8
- How to install Java SE 8, JavaFX 8, NetBeans, and configuring environment variables
- Developing a "Hello World" JavaFX 8 application, and understanding the JavaFX 8 basic application architecture and building blocks

## JavaFX goals

JavaFX came to light with a primary goal – to be used across many types of devices, such as embedded devices, smartphones, TVs, tablet computers, and desktops. JavaFX also follows Java's *write once, run anywhere* paradigm.

JavaFX 8 is written totally from scratch in Java language, it makes you feel at home. Therefore, applications written in JavaFX can be deployed on desktops, laptops, the Web, embedded systems, mobiles, and tablets.

Embedded systems are no longer supported by Oracle; it is left to companies like ARM and others to support it. Mobile devices have never been supported from JavaFX 2.x to 8.x; the support exists now only because of [OpenJFX](https://wiki.openjdk.java.net/display/OpenJFX/Main). The community has benefitted from open source bringing JavaFX to mobile environments.

For more about OpenJFX, visit [https://wiki.openjdk.java.net/display/OpenJFX/Main](https://wiki.openjdk.java.net/display/OpenJFX/Main).

JavaFX is a set of graphics and a media package that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms, in one bundle, without the need for many separate libraries, frameworks, and APIs to achieve the same goal. These separate libraries include media, UI controls, WebView, 3D, and 2D APIs.

So if you are a Java frontend developer, an experienced Java Swing, Flash/Flex, SWT, or web developer looking to take your client-side applications to the next level, and you want to develop an attractive and complex user interface for your customer, then you are on track learning JavaFX skills – this book is for you.
Chapter 1

Getting started

This chapter is an introduction to JavaFX 8; we have already talked about JavaFX 8 as a technology and why you should care about it.

Next, we will navigate its history, exploring its core features and where it could be used.

Before you start using this book to learn JavaFX 8, we will go through the preparation of your development environment by installing various required software bundles, to be able to compile and run many of its examples.

In this chapter, you will learn how to install the required software, such as the Java Development Kit JDK and the NetBeans Integrated Development Environment (IDE).

After installing the required software, you will begin by creating a traditional Hello JavaFX 8 example. Once you feel comfortable with the development environment, as a final verification that we are on the right track, we will walk through the Hello JavaFX 8 source code to understand the basic JavaFX 8 application architecture.

If you are already familiar with the installation of the JDK and the NetBeans IDE, you can skip to Chapter 2, JavaFX 8 Essentials and Creating a custom UI, which covers JavaFX 8 fundamentals and how to create a custom UI component.

So what you are waiting for? Let's get started!

JavaFX history

You might believe JavaFX is quite a new technology, but it actually isn't. JavaFX has been here for a long time; unofficially since 2005. Ever since Sun Microsystems acquired the company SeeBeyond, there has been a graphics-rich scripting language known as F3 (Form Follows Function), which was created by engineer Chris Oliver.

At the JavaOne 2007 conference, Sun Microsystems officially unveiled JavaFX as the language's name instead of F3. During the period 2007 to 2010, Oracle acquired many big companies like BEA Systems, JD Edwards, Siebel Systems, and so on. I was working for Oracle with the responsibility of integrating different customer support channels to the Oracle support website MetaLink, as it was called at that time.
On April 20, 2009, Oracle Corporation announced the acquisition of Sun Microsystems, making Oracle the new steward of JavaFX.

At JavaOne 2010, Oracle announced the JavaFX roadmap, which included its plans to phase out the JavaFX 1.3 scripting language and recreate the JavaFX platform for the Java platform as Java-based APIs. As promised, JavaFX 2.0 SDK was released at JavaOne in October 2011.

In addition to the release of JavaFX 2.0, Oracle took the platform to the next level by announcing its commitment to take steps to make JavaFX open source, thus allowing Java's versatile and strong community to help move the platform forward. Making JavaFX open source increased its adoption, enabled a quicker turnaround time on bug fixes, and generated new enhancements.

Between the versions JavaFX 2.1 and 2.2, the number of new features grew rapidly. JavaFX 2.1 was the official release of the Java SDK on Mac OS. JavaFX 2.2 was the official release of the Java SDK on the Linux operating system.

There was no such thing as JavaFX 3.x, but the big change in the Java development world happened with the Java SE 8 release, which was announced on March 18, 2014. Java SE 8 has many new APIs and language enhancements, which include Lambdas, Stream API, Nashorn JavaScript engine, and JavaFX APIs, which are being incorporated into standard JDK bundles, and JavaFX version becomes 8 as direct successor to JavaFX 2.0.

To see all of the new features in Java SE 8, visit http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html.

When is JavaFX 8 available?

The answer is now. As mentioned before, Java SE 8 was released on March 18, 2014. For developers who use Java to build client-site applications, the JavaFX rich Internet application framework supports Java 8 now.

Most of the Java enterprise edition vendors support Java 8 too. Whether you move to Java SE 8 right away depends on the kind of projects you're working on.

In fact, as outlined in the Oracle JDK Support Roadmap, after April 2015, Oracle will not post further updates of Java SE 7 to its public download sites.
The JavaFX APIs are available as a fully integrated feature of the Java SE Runtime Environment (JRE) and JDK. The JDK is available for all major desktop platforms (Windows, Mac OS X, Solaris, and Linux), therefore JavaFX will also run on all major desktop platforms.

Relating to JavaFX 8, it supports the following APIs:

- 3D graphics
- Rich text support
- Printing APIs.

JavaFX features

The following features are included in JavaFX 8 and later releases as per JavaFX’s official documentation:

- **Java APIs**: JavaFX is a Java library that consists of classes and interfaces that are written in Java code.
- **FXML and Scene Builder**: This is an XML-based declarative markup language for constructing a JavaFX application user interface. You can code in FXML or use JavaFX Scene Builder to interactively design the GUI. Scene Builder generates FXML markup that can be ported to an IDE like NetBeans, where you can add the business logic. Moreover, the FXML file that is generated can be used directly inside the JavaFX application.
- **WebView**: This is a web component that uses WebKit, an HTML render engine technology, to make it possible to embed web pages within a JavaFX application. JavaScript running in WebView can call Java APIs and vice-versa.
- **Swing/SWT interoperability**: The existing Swing and SWT applications can benefit from JavaFX features such as rich graphics, media playback, and embedded web content.
- **Built-in UI controls and CSS**: JavaFX provides all the major UI controls, and some extra uncommon controls like charts, pagination, and accordion that are required to develop a full-featured application. Components can be skinned with standard web technologies such as CSS.
- **3D graphics features**: Support for the 3D graphics library is included.
- **Canvas API**: You can draw directly inside a JavaFX scene area using the Canvas API, which consists of one graphical element (node).
• **Multitouch support**: Multitouch operations are supported based on the capabilities of the underlying platform.

• **Hardware-accelerated graphics pipeline**: JavaFX graphics are based on the graphics-rendering pipeline, *Prism*. The Prism engine smoothly and quickly renders JavaFX graphics when used with a supported graphics card or **graphics processing unit (GPU)**. If a system does not feature one of them, then Prism defaults to the software-rendering stack.

• **High-performance media engine**: This engine provides a stable, low-latency media framework that is based on the *GStreamer* multimedia framework. The playback of web multimedia content is supported with the media pipeline.

• **Self-contained deployment model**: Self-contained application packages have all of the application resources and a private copy of the Java and JavaFX runtimes. They are distributed as native installable packages and provide the same installation and launch experience as native applications for that operating system.

### What’s new in JavaFX 8

The following is a brief summary of the new features and significant product changes made in the JavaFX component of the Java SE 8 release:

- The new *Modena theme* is now the default theme for JavaFX applications.
- Support for additional HTML5 features, including Web Sockets, Web Workers, Web Fonts, and printing capabilities have been added.
- The API enables you to embed **Swing** content into JavaFX applications with the new **SwingNode** class, which improves the Swing interoperability feature.
- **DatePicker**, **Spinner**, and **TableView** built-in UI controls are now available.
- It provides the public JavaFX printing APIs through the **javafx.print** package.
- Support for Hi-DPI displays has been made available.
- CSS-styleable classes became public APIs.
- A scheduled service class has been introduced.
• The 3D graphics library has been enhanced with several new API classes.
• Major updates have been added to the Camera API class in this release.
• Now JavaFX 8 supports rich text capabilities. These include bidirectional and complex text scripts such as Thai and Hindi in UI controls, and multiline, multistyle text in text nodes.
• Dialogs and accessibility APIs are supported.

In Appendix, Become a JavaFX Guru, I have provided a list of all the references (links, books, magazines, articles, blogs, and tools) and real JavaFX 8 production applications you will need to become a JavaFX guru.

The following figure shows the Ensemble8.jar application built using JavaFX 8, showing examples dealing with various JavaFX 8 components, topics and concepts. More interestingly, the source code is available to learn from and modify – consult the last chapter to see how to install this application.
There are many topics covered by the application, especially the new JavaFX 8 3D APIs, which can be found under the Graphics 3D section as seen in the following figure:

![JavaFX 8 3D applications](image)

## Installing the required software

So far, we have had a good introduction to JavaFX and I am as eager as you to start creating and launching our first "Hello JavaFX 8" application. But this can't happen without downloading and installing the right tools that will allow us to create and compile most of the book's code.

You'll need to download and install *Java 8 Java Development Kit (JDK)* or a later version. Not the runtime version (JRE).
Download the latest Java SE 8u45 JDK or higher from the following location:

Download and install NetBeans 8.0.2 or higher from the following link https://netbeans.org/downloads, though the NetBeans IDE All Bundle is recommended, you can use the Java EE bundle as well, as shown in the figure:

<table>
<thead>
<tr>
<th>Supported technologies *</th>
<th>Java SE</th>
<th>Java EE</th>
<th>C/C++</th>
<th>HTML5 &amp; PHP</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetBeans Platform SDK</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Java SE</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Java FX</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Java EE</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Java ME</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>HTML5</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Java Card™ 3 Connected</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>C/C++</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Groovy</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>Bundled servers</td>
<td></td>
<td></td>
<td></td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>GlassFish Server Open Source Edition 4.1</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Apache Tomcat 8.0.15</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

NetBeans bundles download.

Currently, JavaFX 8 runs on the following operating systems:

- Windows OS (XP, Vista, 7, 8) 32- and 64-bit
- Mac OS X (64-bit)
- Linux (32- and 64-bit), Linux ARMv6/7 VFP, HardFP ABI (32-bit)
- Solaris (32- and 64-bit)

**Installing Java SE 8 JDK**

The steps outlined in this section will guide you to successfully download and install Java SE 8. Download the Java SE 8 JDK from the following location:


In the following steps, the Java SE 8u45 JDK 64-bit version (at the time of writing) on the Mac OS X Yosemite (10.10.3) operating system will be used as an example.
The steps are similar on other operating systems and JDK versions. However, if your environment is different, refer to the following link for additional details:

http://docs.oracle.com/javase/8/docs/technotes/guides/install/toc.html

The following are steps to install the Java SE 8 JDK:

1. Install the Java 8 JDK by launching the image file jdk-8u45-macosx-x64.dmg. A screen will appear that looks like the following screenshot once you’ve launched the JDK 8 setup image file. That’s the package setup file. Double-click on it and the installer will launch:

![JDK 8 setup image file](image)

Typically, you will need administrator rights on the machine to install the software.
2. Begin the setup of the Java 8 JDK. The screen in the following screenshot will appear at the beginning of the installation process. Click on the **Continue** button, next on the **Installation** type screen wizard, click on **Install** to begin the installation.

![Java SE Development Kit 8 setup](image1)

3. Once you hit **Install**, you may be asked to supply your password. Supply it, click on **Ok** and the installation will proceed with a progress bar, as shown in following figure:

![Java SE Development Kit 8 installation in progress](image2)

4. The setup will complete the installation of the Java 8 SE Development Kit. click on the **Close** button to exit.
Setting environment variables

Now you need to set a couple of key environment variables. How you set them and the values they should be set to vary depending on your operating system. The two variables to be set are:

- **JAVA_HOME**: This tells your operating system where the Java installation directory is.
- **PATH**: This specifies where the Java executable directory resides. This environment variable lets the system search paths or directories containing executable files. The Java executables reside in the bin directory under the JAVA_HOME home directory.

To make **JAVA_HOME** and **PATH** more permanent, you will want to add them to your system in such a way that they are always made available whenever you boot or log in. Depending on your operating system, you will need to be able to edit environment variable names and values.

In the Windows environment, you can use the keyboard shortcut Windows logo key + Pause/Break key and then click on Advanced system settings to display the Systems Property dialog.

Next, click on Environment Variables. This is where you can add, edit, and delete environment variables. You will add or edit the JAVA_HOME environment variable by using the installed home directory as the value. Shown in this screenshot is the Environment Variables dialog on the Windows operating system:
Let's set the environment variables:

- To set your $JAVA_HOME$ environment variable for the Mac OS X platform, you will need to launch a terminal window to edit your home directory's `.bash_profile` file by adding the following export command:

  ```bash
  export JAVA_HOME=$(/usr/libexec/java_home -v 1.8)
  ```

- On Linux and other Unix operating systems that use Bash shell environments, launch a terminal window and edit either the `~/.bashrc` or `~/.profile` file to contain the export commands:

  ```bash
  export JAVA_HOME=/usr/java/jdk1.8.0
  export PATH=$PATH:$JAVA_HOME/bin
  ```

- On Linux and other Unix operating systems that use c shell (csh) environments, launch a terminal window and edit either the `~/.cshrc` or `~/.login` file to contain the `setenv` commands:

  ```bash
  setenv JAVA_HOME /usr/java/jdk1.8.0_45
  setenv PATH ${JAVA_HOME}/bin:${PATH}
  ```

Once you've set up your path and the $JAVA_HOME$ environment variables, you will want to verify your setup by launching a terminal window and executing the following two commands from the command prompt:

```
java -version
javac -version
```

The output in each case should display a message indicating the Java SE 8 version of the language and runtime.

### Installing the NetBeans IDE

When developing JavaFX applications, you will be using the NetBeans IDE (or any other IDE of your preference). Be sure to download the correct NetBeans version containing JavaFX. To install the NetBeans IDE, follow these steps:

1. Download the NetBeans IDE 8.0.2 or later from the following location:

2. Launch the .dmg image file `netbeans-8.0.2-macosx.dmg`. The image will be verified and a folder containing the installer package archive, `netbeans-8.0.2.pkg`, will open; double-click on it to launch the installer. A dialog box will appear with the message: *This package will run a program to determine if the software can be installed.* Click on the **Continue** button.

3. Once you've launched the NetBeans installation dialog, click on **continue** again. Next, accept the license and click on **Continue** and then on **Agree**.

4. Click on the **Install** button to proceed. The following screenshot shows a **Mac** security warning prompt; supply your password and click on **Install Software**.

![The Mac Security Warning dialog](image)

The Mac Security Warning dialog
5. The NetBeans IDE installation processes will begin. The following screenshot shows the installation progress bar:

![Installation progress](image)

6. Click on the Close button to complete the installation, as shown here:

![Setup complete](image)

Now you are ready to move on and create JavaFX applications.
Creating "Hello World" JavaFX-style applications

The best way to show you what it is like to create and build a JavaFX application would be with a Hello World application.

In this section, you will be using the NetBeans IDE we just installed to develop, compile, and run a JavaFX-based Hello World application.

Using the Netbeans IDE

To quickly get started with creating, coding, compiling, and running a simple JavaFX-style Hello World application using the NetBeans IDE, follow the steps outlined in this section:

1. From the File menu, choose New Project.
2. From JavaFX application category, choose JavaFX Application. Click on Next.
3. Name the project HelloJavaFX. Optionally, you can define the package structure for application classes. Then click on Finish as shown in the following screenshot:

![New JavaFX application wizard](image.png)
NetBeans opens the HelloJavaFX.java file and populates it with the code for a basic "Hello World" application.

You will find that this version of code has been modified a bit from the one NetBeans actually creates, and you can compare them to find differences, but they have the same structure. I did that to show the result on the text node on the Scene instead of the console when clicking on the Say 'Hello World' button. For that, a VBox container has also been used.

4. Right-click on the project and click on Run from the menu as shown here:

![Running the application]

5. NetBeans will compile and run the application. The output should be as shown in the following screenshot:

![JavaFX Hello World launched from the NetBeans IDE]
6. Click on the button and you should see the following result:

Here is the modified code of the basic Hello world application (HelloJavaFX.java):

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import static javafx.geometry.Pos.CENTER;
import javafx.scene.layout.VBox;

/**
 * @author mohamed_taman
 */
public class HelloJavaFX extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        Text message = new Text();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(event -> {
            message.setText("Hello World! JavaFX style :)");
        });
```
Chapter 1

```java
VBox root = new VBox(10, btn, message);
root.setAlignment(CENTER);

Scene scene = new Scene(root, 300, 250);

primaryStage.setTitle("Hello JavaFX 8 World!");
primaryStage.setScene(scene);
primaryStage.show();
```

```java
public static void main(String[] args) {
    launch(args);
}
```

**How it works**

Here are the important things to know about the basic structure of a JavaFX application:

- The main class for a JavaFX application should extend the `javafx.application.Application` class. The `start()` method is the main entry point for all JavaFX applications.

- A JavaFX application defines the user interface container by means of a `stage` and a `scene`. The JavaFX `Stage` class is the top-level JavaFX container. The JavaFX `Scene` class is the container for all content. The following code snippet creates a stage and scene and makes the scene visible in a given pixel size - `new Scene(root, 300, 250)`.

- In JavaFX, the content of the scene is represented as a hierarchical scene graph of nodes. In this example, the root node is a `VBox` layout object, which is a resizable layout node. This means that the root node's size tracks the scene's size and changes when a user resizes the stage.

- The `VBox` is used here as the container that arranges its content nodes vertically in a single column with multiple rows. We have added the button `btn` control to the first row in the column, then the text `message` control to the second row on the same column, with vertical space of 10 pixels, as in the following code snippet:

  ```java
  VBox root = new VBox(10, btn, message);
  root.setAlignment(CENTER);
  ```
• We set the button control with text, plus an event handler to set the message text control to **Hello World! JavaFX style :)** when the button is clicked on.

• You might note a strange code syntax written in Java, with no compiler errors. This is a **Lambda** expression, which has been added to Java SE 8, and we are going to talk about it briefly in Chapter 2, *JavaFX 8 Essentials and Creating a custom UI*. With a slight comparison to old anonymous inner classes style, it is cleaner and more concise to use Lambda expression now. Have a look at this comparison of code:

Old School:

```java
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        message.setText("Hello World! JavaFX style :)");
    }
});
```

New Era:

```java
btn.setOnAction(event -> {
    message.setText("Hello World! JavaFX style :)");
});
```

• The **main()** method is not required for JavaFX applications when the **JAR** file for the application is created with the JavaFX Packager tool, which embeds the JavaFX Launcher in the JAR file.

• However, it is useful to include the **main()** method so you can run JAR files that were created without the JavaFX Launcher, such as when using an IDE in which the JavaFX tools are not fully integrated. Also, **Swing** applications that embed JavaFX code require the **main()** method.

• Here, in our **main()** method's entry point, we launch the JavaFX application by simply passing in the command-line arguments to the **Application.launch()** method.

• After the **Application.launch()** method has executed, the application will enter a ready state and the framework internals will invoke the **start()** method to begin.

• At this point, the program execution occurs on the **JavaFX application thread** and not on the **main thread**. When the **start()** method is invoked, a JavaFX **javafx.stage.Stage** object is available for you to use and manipulate.
Advanced topics will be discussed at length in the next chapters. More importantly, we will go through the JavaFX application thread in the coming chapters. In the last three chapters, we will see how to bring the result from other threads into the JavaFX application thread in order to render it correctly on the scene.

**Summary**

So far, you have learned what JavaFX is and seen its power. You have managed to download and install both Java 8 JDK and NetBeans IDE. After successfully installing the prerequisite software, you created a JavaFX Hello World GUI application through the NetBeans IDE. After learning how to compile and run a JavaFX application, you did a quick code walkthrough of the source file HelloJavaFX.java.

Next, in Chapter 2, *JavaFX 8 Essentials and Creating a custom* you'll learn about JavaFX 8 architecture components and engines, which allow JavaFX applications to run efficiently and smoothly under the hood. You will also learn about the most common layout UI components and get an idea about theming your application as a whole or as individual scene nodes.

We will also be covering Java SE 8's most important feature, Lambda expressions, and how it works. We will then get insights into Scene Builder as a declarative UI and productive tool, then learn about generated FXML-based markup document and how to import it into NetBeans IDE to continue your application logic implementation to associate it to the already declared UI controls inside your FXML document.

Finally, you will be able to create a custom UI component that isn't bundled with default JavaFX 8 UI controls.
Where to buy this book

You can buy JavaFX Essentials from the [Packt Publishing website](http://www.PacktPub.com).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

[Click here](http://www.PacktPub.com) for ordering and shipping details.