Introduction to FXML and JavaFX CSS

1. XML Tutorial

http://www.w3school.com.cn/xml/index.asp

1) Introduction
   http://www.w3school.com.cn/xml/xml_intro.asp
2) How to use
   http://www.w3school.com.cn/xml/xml_usedfor.asp
3) XML Tree
   http://www.w3school.com.cn/xml/xml_tree.asp
4) XML Syntax
   http://www.w3school.com.cn/xml/xml_syntax.asp
5) XML Elements
   http://www.w3school.com.cn/xml/xml_elements.asp
6) XML Attributes
   http://www.w3school.com.cn/xml/xml_attributes.asp
7) XML Namespaces
   http://www.w3school.com.cn/xml/xml_namespaces.asp
8) Encoding
   http://www.w3school.com.cn/xml/xml_encoding.asp
9) XML Based Technologies
   http://www.w3school.com.cn/xml/xml_technologies.asp
10) XML Examples
   http://www.w3school.com.cn/xml/xml_real_life.asp
   http://www.w3school.com.cn/example/xmle_examples.asp

2. CSS Tutorial

http://www.w3school.com.cn/css/index.asp

1) Introduction
   http://www.w3school.com.cn/css/css_jianjie.asp
2) CSS Syntax
   http://www.w3school.com.cn/css/css_syntax.asp
   http://www.w3school.com.cn/css/css_syntax_pro.asp

3) Syntax Descendant Selector
   http://www.w3school.com.cn/css/css_syntax_descendant_selector.asp
4) Syntax ID Selector
   http://www.w3school.com.cn/css/css_syntax_id_selector.asp
5) Syntax Class Selector
   http://www.w3school.com.cn/css/css_syntax_class_selector.asp
6) How to Construct
   http://www.w3school.com.cn/css/css_howto.asp
7) XML CSS
   http://www.w3school.com.cn/xml/xml_display.asp
   http://www.w3school.com.cn/css/css_selector_type.asp

3. FXML

FXML is a scriptable, XML-based markup language for constructing Java object graphs. It provides a convenient alternative to constructing such graphs in procedural code, and is ideally suited to defining the user interface of a JavaFX application, since the hierarchical structure of an XML document closely parallels the structure of the JavaFX scene graph. The extension filename of FXML is fxml, ex: myLayout.fxml.

In FXML, an XML element represents one of the following:

- A class instance
- A property of a class instance
- A "static" property
- A "define" block
- A block of script code

1) Importing a class is done using the "import" processing instruction (PI).

```
<?import javafx.scene.control.Label?>
```

The following PI imports the javafx.scene.control.Label class into the current FXML document's namespace.

```
<?import javafx.scene.control.*?>
```

This PI imports all classes from the javafx.scene.control package into the current namespace.

```
<?import javafx.scene.control.Label?>
<Label text="Hello, World!"/>
```

This is a simple but complete example that creates an instance of javafx.scene.control.Label and

sets its "text" property to "Hello, World!":

2) Property Setters

The following FXML creates an instance of the Label class and sets the value of the label's "text" property to "Hello, World!":

```
<?import javafx.scene.control.Label?>
<Label>
    <text>Hello, World!</text>
</Label>
```

This produces the same result as the earlier example which used an attribute to set the "text" property:

```
<?import javafx.scene.control.Label?>
<Label text="Hello, World!"/>
```

The properties of the elements are the method and fields of class, and you can search in following URL:
https://openjfx.io/javadoc/11/index.html

You can refer to following webpage for more FXML syntax information.
https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html

4. JavaFX CSS

JavaFX Cascading Style Sheets (CSS) is based on the W3C CSS version 2.1 with some additions from current work on version 3. JavaFX CSS also has some extensions to CSS in support of specific JavaFX features. The goal for JavaFX CSS is to allow web developers already familiar with CSS for HTML to use CSS to customize and develop themes for JavaFX controls and scene graph objects in a natural way. The JavaFX CSS support and extensions have been designed to allow JavaFX CSS style sheets to be parsed cleanly by any compliant CSS parser, even though it might not support JavaFX extensions. This enables the mixing of CSS styles for JavaFX and for other purposes (such as for HTML pages) into a single style sheet. To this end, all JavaFX property names have been prefixed with a vendor extension of "-fx-". Even properties that might seem to be compatible with standard HTML CSS have been prefixed, because JavaFX has somewhat different semantics for their values.

JavaFX CSS does not support CSS layout properties such as float, position, overflow, and width. However, the CSS padding and margins properties are supported on some JavaFX scene graph objects. All other aspects of layout are handled programmatically in JavaFX code. In addition, CSS support for HTML-specific elements such as Tables are not supported since there is no equivalent construct in JavaFX. The extension filename of CSS is css, ex: myLayout.css.

The following example shows how to link the Label element in FXML to a CSS file.

```
<Label stylesheets="@LabelCSS.css">
```

1) id

The id selector you use to set a CSS ID to component in fxml, for example

```
<Text id="welcome-text" .../>
```

and in stylesheet (css file) you have something like

```
#welcome-text { font-size: 16pt; }
```

, so this will be applied to your Text.

The fx:id you use in fxml, for example

```
<Text fx:id="welcome-text" .../>
```

If you want to work with components in your controller class with java code, you have to annotate them with

```
@FXML
Text myWelcomeText
```

In fact, you only need to use fx:id in fxml, and it can be used both in Controller in java code and CSS stylesheet. But if you use fx:id and id at the same component, CSS stylesheet follows id.

2) class

In JavaFX CSS, you have to use styleClass, not “Class” in general CSS.

```
<Label styleClass="label1"/>
```

And in stylesheet (css file) you have something like

```
.label1 {  -fx-text-fill: red; }
```

You can refer to following webpage for more information of JavaFX CSS.
https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

5. How to load and run FXML in JavaFX

```
1 import javafx.application.Application;
2 import javafx.fxml.FXMLLoader;
3 import javafx.scene.Parent;
4 import javafx.scene.Scene;
5 import javafx.stage.Stage;
6
7 public class FontCSS extends Application {
```

```
8     @Override
9     public void start(Stage stage) throws Exception {
10        // loads FontCSS.fxml and confiugres the Draw
11        Parent root = FXMLLoader.load(getClass().getResource("FontCSS.fxml"));
12
13        Scene scene = new Scene(root); // attach scene graph to scene
14        stage.setTitle("Draw Stars");  // displayed in window's title bar
15        stage.setScene(scene); // attach scene to stage
16        stage.show(); // display the stage
17    }
18
19    // application execution starts here
20    public static void main(String[] args) {
21        launch(args); // create a FontCSS object and call its start method
22    }
23 }
```

3) Class Application inherits methods of Object and has getClass().getResource() method which can return URL of source.
4) Class FXMLLoader has method to load an object hierarchy from an XML document. Therefore, you can use load() method in Class javafx.fxml.FXMLLoader to load .fxml file in JavaFX. FXMLLoader::load(URL loation) throws IOException:Loads an object hierarchy from a FXML document.
5) In general, the code is

```
(User Interaction Component Class) xxx =
FXMLLoader.load(getClass().getResource("FXML_file.fxml"));
```

6) In this example, the (User Interaction Component Class) is Class Parent. Class Parent is the base class for all nodes that have children in the scene graph.

You can follow below instructions to compile and run the code with FXML.

1) Compile

Input

Javac --module-path $env:PATH_TO_FX --add-modules=javafx.controls,javafx.fxml BasicShapes.java

to compile the code with fxml, and you can find it generates HelloFX.class in the same folder.

2) Run

Input

Java --module-path $env:PATH_TO_FX --add-modules=javafx.controls,javafx.fxml BasicShapes

and you will see execution result.

6. Examples

1) Fonts

FontCSS.java

```
// FontCSS.java
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class FontCSS extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        // loads FontCSS.fxml and confiugres the Drwa
        Parent root = FXMLLoader.load(getClass().getResource("FontCSS.fxml"));

        Scene scene = new Scene(root); // attach scene graph ot scene
        stage.setTitle("Show Fonts");  // displayed in window's title bar
        stage.setScene(scene); // attach scene to stage
        stage.show(); // display the stage
    }

    // application execution starts here
    public static void main(String[] args) {
        launch(args); // create a FontCSS object and call its start method
    }
}
```

FontCSS.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- FontCSS.fxml -->
<!-- FontCSS GUI that is styled via external CSS -->

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<VBox styleClass="vbox" stylesheets="@FontCSS.css"
```

```
   xmlns="http://javafx.com/javafx/8.0.60"
   xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Label fx:id="label1" text="Arial 14pt bold" />
      <Label fx:id="label2" text="Times New Roman 16pt plain" />
      <Label fx:id="label3" text="Courier New 16pt bold and italic" />
      <Label fx:id="label4" text="Default font 14pt with underline" />
      <Label fx:id="label5" text="Default font 14pt with strikethrough" />
   </children>
</VBox>
```

FontCSS

```
/* FontsCSS.css */
/* CSS rules that style the VBox and Labels */

.vbox {
   -fx-spacing: 10;
   -fx-padding: 10;
}

#label1 {
   -fx-font: bold 14pt Arial;
   -fx-text-fill: red;
}

#label2 {
   -fx-font: 16pt "Times New Roman";
}

#label3 {
   -fx-font: bold italic 16pt "Courier New";
}

#label4 {
   -fx-font-size: 14pt;
   -fx-underline: true;
   -fx-text-fill: green;
}

#label5 {
```

```
   -fx-font-size: 14pt;
}

#label5 .text {
   -fx-strikethrough: true;
}
```

2) Basic Shapes

BasicShapes.java

```
// BasicShapes.java
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class BasicShapes extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        Parent root =
                  FXMLLoader.load(getClass().getResource("BasicShapes.fxml"));
        Scene scene = new Scene(root);
        stage.setTitle("Basic Shapes");
        stage.setScene(scene);
        stage.show();
    }

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

BasicShapes.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- BasicShapes.fxml -->
<!-- Defining Shape objects and styling via CSS -->

<?import javafx.scene.layout.Pane?>
```

```
<?import javafx.scene.shape.Arc?>
<?import javafx.scene.shape.Circle?>
<?import javafx.scene.shape.Ellipse?>
<?import javafx.scene.shape.Line?>
<?import javafx.scene.shape.Rectangle?>

<Pane id="Pane" prefHeight="110.0" prefWidth="630.0"
   stylesheets="@BasicShapes.css" xmlns="http://javafx.com/javafx/8.0.60"
   xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Line fx:id="line1" endX="100.0" endY="100.0"
         startX="10.0" startY="10.0" />
      <Line fx:id="line2" endX="10.0" endY="100.0"
         startX="100.0" startY="10.0" />
      <Rectangle fx:id="rectangle" height="90.0" layoutX="120.0"
         layoutY="10.0" width="90.0" />
      <Circle fx:id="circle" centerX="270.0" centerY="55.0"
         radius="45.0" />
      <Ellipse fx:id="ellipse" centerX="430.0" centerY="55.0"
         radiusX="100.0" radiusY="45.0" />
      <Arc fx:id="arc" centerX="590.0" centerY="55.0" length="270.0"
         radiusX="45.0" radiusY="45.0" startAngle="45.0" type="ROUND" />
   </children>
</Pane>
```

BasicShpes.css

```
/* BasicShapes.css */
/* CSS that styles various two-dimensional shapes */

Line, Rectangle, Circle, Ellipse, Arc {
    -fx-stroke-width: 10;
}

#line1 {
    -fx-stroke: red;
}

#line2 {
    -fx-stroke: rgba(0%, 50%, 0%, 0.5);
    -fx-stroke-line-cap: round;
```

```
}

Rectangle {
   -fx-stroke: red;
   -fx-arc-width: 50;
   -fx-arc-height: 50;
   -fx-fill: yellow;
}

Circle {
   -fx-stroke: blue;
   -fx-fill: radial-gradient(center 50% 50%, radius 60%, white, red);
}

Ellipse {
   -fx-stroke: green;
   -fx-fill: image-pattern("yellowflowers.png");
}

Arc {
   -fx-stroke: purple;
   -fx-fill: linear-gradient(to right, cyan, white);
}
```