> 由 KiOii (*_EM_Cpper*)整理。 （KiOii (*_EM_Cpper*) makes this note.）

Styling Nodes

Styling Nodes

In this chapter

- 什么是CSS (cascading style sheeets)
- styles, skins, themes 的区别
- JavaFX中的CSS的命名约定
- 如何给scene添加CSS
- 如何在JavaFX application中use and override 默认的样式表

- 如何为Node添加 inline style
- 关于不同类型的 CSS 属性
- 关于 CSS selectors
- 如果通过CSS selectors 查看 scene graph中的 Node
- 如何使用编好的 样式表

1. What Is a Cascading Style Sheet?

- define : a language
- 功能定位：描述 GUI应用程序中 UI elements 的表现（look or style)
- 主要用途：描述web中的 HTML style elements
    - 特点：允许将content和behavior的表现分离开来
- JavaFX中
    - 允许使用 CSS 定义 style
    - 你可以使用 JavaFX 类库 或者 FXML 去定义 UI elements，然后用CSS去定义它们的style
    - CSS提供了编写规则（rules）去设置 visual 属性
        - **rules** : **selector** and **a set of** `property-value` **pairs**
            - **selector**
                - type : String
                - 作用：标识被rules应用的UI elements
            - `property-value pair`
                - 组成：name and 由冒号 `:` 分隔的相应值
            - pair分割符：`;`
            - pair集合: 由 `{}` 包裹

                > 位于 selector 之后

            - example

```
.button{
    -fx-background-color:red;
    -fx-text-fill: white;
    // :  分割了 属性和值
    // ;  分割了 pairs
    // .button is selector : 指定了all buttons
}
```

2. What are Styles, Skins, and Themes?

- A CSS style : a style
- A collection of CSS style : a style sheet

Styles

- 样式提供了分离UI元素的表示和内容的机制

- 促进了 visual 属性和 值的分块，被多个UI elements共享

Skins

- a collection of styles （ all styles)
- 定义了应用程序的表现
- **Skinning** : 改变应用程序（或skin）外观的**过程**
  - JavaFX 没有为 skinning提供特别的机制
  - 通过使用JavaFX CSS和 JavaFX API

    > 但是，使用可用于Scene类和其他与UI相关的类的JavaFX CSS和JavaFX API，您可以轻松地为JavaFX应用程序提供 skinning

Themes

- 操作系统反映在所有应用程序的UI元素外观中的视觉(visual)特征
- skins : 应用程序指定
- themes : 操作系统指定
- 当前的主题(theme)被修改时，你将通过改变skin去匹配theme
  - JavaFX 不直接支持 themes

3. A Quick Example

> 在JavaFX中，你可以为所有的button设置背景颜色和文本颜色

- example

  ```
  .button{
      -fx-background-color:red;
      -fx-text-fill:white;
  }
  ```

  > 保存为 buttonstyles.css 文件
  >
  > 放置于 resources\css 文件目录下

  - 保存文件后放置于文件夹中
- scene对象 包含一个 关于style sheets URLs的ObservableList
  - 获取该ObservableList的引用： **scene . getStylesheets()**
    - easy example

      ```
      Scene scene;
      ...
      scene.getStylesheets().add("resources/css/buttonStyles.css");
      ```

    - example

      ```

      package com.javafx.style;
      ```

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.Group;

public class ButtonStyleTest extends Application {
    public static void main(String[] args) {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage) {
        Button yesBtn = new Button("Yes");
        Button noBtn = new Button("No");
        Button cancelBtn = new Button("Cancel");
        Group root = new Group();
        root.getChildren().addAll(yesBtn, noBtn, cancelBtn);
        Scene scene = new Scene(root);
        // Add a style sheet to the scene
        scene.getStylesheets().add("CSS/buttonStyles.css");
        stage.setScene(scene);
        stage.setTitle("Styling Buttons");
        stage.show();
    }
}
```

4. Nameing Conventions in JavaFX CSS

- JavaFX对CSS样式类(style class)和属性使用略有不同的命名约定
- **CSS style class names** : **lowercased** (全部小写,可能带 -)
  - example :
    - 对于Button class，button是style class名字
    - 对于TextField class, text-field是style class名字

> 懂得JavaFX class 与 CSS style class之间的区别是很重要的

  - A CSS style class is used **as a selector** in a style sheet : 例如之前写的 button
- **Property names** in JavaFX style : `-fx-` 开头
  - example
    - 属性名：font-size
      - CSS styles : `font-size`
      - JavaFX CSS style : `-fx-font-size`
  - JavaFX使用约定将style property 名称映射到实例变量
    - example
      - 实例变量：textAlignment
      - style property : -fx-text-alignment

> 一幕了然的映射关系

5. Adding Style Sheets

> 你可以增加多个 style sheets 到JavaFX应用程序中

- **Style sheet** are added to a scene or parents
- Scene和Parent类维护一个observable 样式表的字符串URL

  > 我们之前提到的 getStylesheets 和 ObservableList

- example

```
// 增加两个 style sheets 到 scene
Scene scene;
scene.getStylesheets().addAll("CSS/ss1.css","CSS/ss2.css");
// 增加一个 style sheet 到 VBox对象
VBox root = new VBox();
root.getStylesheets().add("vbox.css");
```

- 该路径是如何被解析的?
  - 相对 URL : `resource/css/ss1.css`
  - 绝对 URL : `/resource/css/ss1.css`
  - 绝对 URL : `http://jdojo.com/resources/css/ss1.css`
  - 绝对 URL : `file:/C:/css/ss2.css`
- 如何发现无法访问你的 style sheet，那么你可以使用其他方法试试

```
Scene scene;
...
String urlString = Test.class.getClassLoader()
.getResource("resources/css/hjfx.css")
.toExternalForm();
scene.getStylesheets().add(urlString);
```

6. Default Style Sheet

> JavaFX runtime was always using a style sheet behind the scenes
>
> 该style sheet名为 Modena.css

- `Modena.css`
  - 被称为：default style sheet 或者 user-agent style sheet
  - 文件位置：`jfxrt.jar`

  > java8之前是 caspian.css

- 关于 Application 类定义的两个字符串常量： 表示两种主题
  - STYLESHEET_CASPIAN

    - STYLESHEET_MODENA
- `public static void setUserAgenStylesheet(String url)`
    - set an application-wide default
    - 当url为 null时，恢复平台默认样式表，如：Modena (java8), Caspian (旧版)

    ```
    Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);
    ```

- `public static String getUserAgentStylesheet()`
    - return default style sheet,如果是默认的，则返回 null

7. Adding Inline Styles

- scene graph中的节点的CSS styles可能来自 style sheet或者 一个 inline style

    > 前面我们已经介绍了如何 add css文件
    >
    > 现在，我们介绍如何为节点添加 inline style

- **Node** class
    - **StringProerty** : style
        - 保存了node的 inline style
    - **setStyle (String inlineStyle)** : set inline style of a node
    - **getStyle** : get inline style of a node
- style 、style sheet、inline style 的区别
    - style : style sheet中的style包含一个selector 和 包裹起来的 property-value pairs

        > 作用于 一个或者多个节点，或者不作用于节点

        - selector 决定了影响到哪些node的
    - inline style : 不包含 selector, 只包含 property-value pairs

        > 因为他只作用于调用setStyle的对象

        ```
        Button yesBtn = new Button("Yes");
        yesBtn.setStyle("-fx-text-fill:red; -fx-font-weight:bold;");
        ```

- example

    ```
    package com.javafx.style;

    import javafx.application.Application;
    import javafx.geometry.Insets;
    import javafx.scene.Scene;
    import javafx.scene.control.Button;
    import javafx.scene.layout.HBox;
    import javafx.scene.layout.VBox;

    import javafx.stage.Stage;
    ```

```java
public class InlineStyles extends Application {
public static void main(String[] args) {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage) {
        Button yesBtn = new Button("Yes");
        Button noBtn = new Button("No");
        Button cancelBtn = new Button("Cancel");
        // Add an inline style to the Yes button
        yesBtn.setStyle("-fx-text-fill: red; -fx-font-weight: bold;");

        Button openBtn = new Button("Open");
        Button saveBtn = new Button("Save");
        Button closeBtn = new Button("Close");
        VBox vb1 = new VBox();
        vb1.setPadding(new Insets(10, 10, 10, 10));
        vb1.getChildren().addAll(yesBtn, noBtn, cancelBtn);
        VBox vb2 = new VBox();
        vb2.setPadding(new Insets(10, 10, 10, 10));
        vb2.getChildren().addAll(openBtn, saveBtn, closeBtn);
        // Add a border to VBoxes using an inline style
        vb1.setStyle("-fx-border-width: 4.0; -fx-border-color: blue;");
        vb2.setStyle("-fx-border-width: 4.0; -fx-border-color: blue;");
        HBox root = new HBox();
        root.setSpacing(20);
        root.setPadding(new Insets(10, 10, 10, 10));
        root.getChildren().addAll(vb1, vb2);
        // Add a border to the HBox using an inline style
        root.setStyle("-fx-border-width: 10.0; -fx-border-color: navy;");
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Using Inline Styles");
        stage.show();
    }
}
```

8. Priorities of Styles for a Node

- 在JavaFX应用程序中，节点的可视属性(visual property)可能来自多个来源，并且非常普遍
  - JavaFX使用一个rule去决定使用那个来源

```
Button yesBtn = new Button("Yes");
yesBtn.setStyle("-fx-font-size: 16px");
yesBtn.setFont(new Font(10));
Scene scene = new Scene(yesBtn);
scene.getStylesheets().addAll("resources/css/stylespriorities.css");
...

// 关于上面指定的css文件
.button {
    -fx-font-size: 24px;
    -fx-font-weight: bold;
}
```

- 那么问题来了，字体有默认大小，setFont设置了字体大小，css文件也设置了字体大小，inline style 设置了字体大小，那么字体大小是多少
    - JavaFX使用的优先级规则
        - Inline style (最高)
        - Parent style sheets
        - Scene style sheets
        - Value set in the code using JavaFX API (setFont等)
        - Users agent style sheets (默认代理窗口的 style sheets)

9. Inheriting CSS Properties

- JavaFX为CSS属性提供了两种类型的继承
    - Inheritance of CSS property types
        - 描述属性的类被子类继承属性
        - example
            - Node 类有一个 cursor属性，这样映射的 style property就是 `-fx-cursor` ,由于Node是基类，因此所有继承自Node类都有了 `-fx-cursor` CSS property及值
    - Inheritance of CSS property values
        - a CSS property for a node may inherit its value from its parent
            - 如果属性是默认继承的，那么就什么都不用做 (不必指定 属性值为 inherit)
            - 如果想重写继承的值，那么需要显式指定
- example

```
/* Parent Node (HBox)*/
-fx-cursor: hand;             // 该属性默认被继承
-fx-border-color: blue;
-fx-border-width: 5px;
/* Child Node (OK Button)*/
-fx-border-color: red;        // override
-fx-border-width: inherit;    // inherit
```

> cursor : 默认继承
>
> textAlignment：默认继承

> font： 默认继承
>
> Border-related： 没有被默认继承

10. Types of CSS Properties

- CSS style property 的值也有类型
    - inherit
    - boolean
    - string
    - number
    - angle
    - point
    - color-stop
    - URI
    - effect
    - font
    - paint

> 这些跟 java没有关系 而是 CSS types

> 在将它们分配给节点之前，JavaFX运行时负责解析并将这些类型转换为适当的JavaFX类型

10.1 The inherit Type

- `-fx-border-width: inherit; // inherit`

  > 用于继承基类值

10.2 The string Type

```
.my-control{
    -fx-skin:"com.jdojo.MySkin";
    -fx-font:normal bold 20px 'serif';
}
```

> 字符串值可以用单引号或双引号括起来。 如果字符串值用双引号括起来，则作为该值的一部分的双引号应该被转义，例如\“或\ 22。同样，单引号作为单引号中包含的字符串值的一部分必须转义，例如 作为\'或\ 27
>
> 字符串值不能直接包含换行符。 要在字符串值中嵌入换行符，请使用转义序列\ A或\ 00000a

10.3 The number Type

```
.my-style{
    -fx-opacity:0.60;
}
```

> 表示大小的CSS属性的值可以使用以下单位长度的数字来指定。 长度单位可以是px（像素），mm（毫米），cm（厘米），in（英寸），pt（点），pc（皮卡），em或ex。 还可以使用长度的百分比来指定大小，例如节点的宽度或高度。 如果指定了百分比的单位，它必须紧跟在数字后面，例如12px，2em，80%

```
.my-style{
    -fx-font-size:12px;
    -fx-background-radius:0.5em;
    -fx-border-with:5%;
}
```

10.4 The angle Type

> 使用数字和单位指定角度。 角度的单位可以是deg（度），rad（弧度），grad（梯度）或turn（turn）

```
.my-style{
    -fx-rotate:45deg;
}
```

10.5 The point Type

> 一个点使用x和y坐标指定。 可以使用由空格分隔的两个数字指定它，例如，0 0，100，0，90 67或百分比形式，例如2%2%

```
.my-style{
    -fx-background-color:linear-gradient(from 0 0 to 100 0,repeat,red,blue);
}
```

10.6 The color-stop Type

> 用于 gradient 中
>
> 包含color and stop distance

> white 0%, yellow 50%, yellow 100px

10.7 The URI Type

> `url ( <address> )`

```
.image-view{
    -fx-image:url("http://jdojo.com/myimage.png");
}
```

10.8 The effect Type

- CSS function
  - `dropshadow(<blur-type>,<color>,<radius>,<spread>,<x-offset>,<y-offset>)`
  - `innershadow(<blur-type>,<color>,<radius>,<choke>,<x-offset>,<y-offset>)`
  - `<blur-type>` : Gaussian, one-pass-box, three-pass-box, two-pass-box
  - `<color>` :
  - `<radius>` : 指出了 radius of the shadow blur kernel （0.0 ~ 127.0)
  - `spread / choke` ： 0.0 ~ 1.0

```
.drop-shadow-1{
    -fx-effect:dropshadow(gaussian,gray,10,0.6,10,10);
}
.drop-shadow-2{
    -fx-effect:dropshadow(one-pass-box,gray,10,0,6,10,10);
}
.inner-shadow-1{
    -fx-effect:innershadow(gaussian,gray,10,0.6,10,10);
}
```

10.9 The font Type

- 包含四个 attribute(属性)
  - family
  - size
  - style
  - weight
- 两者指定 font CSS property的方式
  - 分开
    - `-fx-font-family`

      > string value
      >
      > "Arial"
      >
      > "Times"
      >
      > "serif"
      >
      > "sans-serif"
      >
      > "monospace"
      >
      > ...

    - `-fx-font-size`

      > px,em,pt,in,cm为单位

    - `-fx-font-style`

> normal
>
> italic 斜体
>
> oblique 倾斜

- `-fx-font-weight`

> normal
>
> bold
>
> bolder
>
> lighter
>
> 100
>
> 200
>
> 300
>
> 400
>
> 500
>
> 600
>
> 700
>
> 800
>
> 900

```
.my-font-style{
    -fx-font-family:"serif";
    -fx-font-size:20px;
    -fx-font-style:normal;
    -fx-font-weight:bolder;
}
```

- 由 `-fx-font` 统一指定：`-fx-font:<style> <weight> <size> <family>`

```
.my-font-style{
    -fx-font:italic bolder 20px "serif";
}
```

10.10 The paint Type

> 指定颜色：填充矩形的颜色，按钮的背景颜色等等

- Using linear-gradient
- Using radial-gradient

```
.my-style {
-fx-fill: linear-gradient(from 0% 0% to 100% 0%, black 0%, red 100%);
-fx-background-color: radial-gradient(radius 100%, black, red);
}
```

- Using various color value and color function
- 指定solid color
  - Using named colors

    ```
    .my-style {
    -fx-background-color: red;
    }
    ```

  - Using looked-up colors

    ```
    .root {
    my-color: black;
    }
    .my-style {
    -fx-fill: my-color;
    }
    ```

  - Using the rgb and rgba function

    ```
    .my-style-1 {
    -fx-fill: rgb(0, 0, 255);
    }
    .my-style-2 {
    -fx-fill: rgba(0, 0, 255, 0.5);
    }
    ```

  - Using red, green, blue(RGB) hexadecimal notaion

    ```
    .my-style-1 {
    -fx-fill: #0000ff; // #RRGGBB
    }
    .my-style-2 {
    -fx-fill: #0bc;     // #RGB
    }
    ```

  - Using the hsb or hsba funtion

    > hsb(hue, saturation, brightness) or hsba(hue, saturation, brightness, alpha)

```
.my-style-1 {
-fx-fill: hsb(200, 70%, 40%);
}
.my-style-2 {
-fx-fill: hsba(200, 70%, 40%, 0.30);
}
```

- Using color function : derive and ladder

  > 从其他颜色中计算颜色

  > derive(color, brightness) brightness : -100% ~ 100%

  ```
  .my-style {
  -fx-fill: derive(red, -20%);
  }
  ```

  > ladder(color, color-stop-1, color-stop-2, ...)
  >
  > 函数想象为使用颜色停止创建渐变，然后使用指定颜色的亮度返回颜色值。 如果指定颜色的亮度为x%，则将返回距渐变起点x%距离处的颜色。 例如，对于0%亮度，将返回渐变0.0端的颜色; 对于40%的亮度，返回梯度0.4端的颜色

  ```
  .root {
  my-base-text-color: red;
  }
  .my-style {
  -fx-text-fill: ladder(my-base-text-color, white 29%, black 30%);
  }
  ```

11. Specifying Background Colors

> p 241

- `-fx-background-color`
- `-fx-background-radius`
- `-fx-background-insets`

12. Specifying Borders

> p242

- `-fx-border-color`
- `-fx-border-width`
- `-fx-border-radius`
- `-fx-border-insets`
- `-fx-border-style`

13. Understanding Style Selectors

- **types of selectors**
  - class selectors（类选择器）
  - pseudo-class selectors（伪类选择器）
  - ID selectors （ID选择器）
  - others (其他)

13.1 Using Class Selectors

- styleClass
  - class : **Node**
  - type : `ObservablleList<String>`
  - example

    ```
    HBox hb = new HBox();
    hb.getStyleClass().addAll("hbox","mybox");
    ```

    - A style sheet with two style class selector

      ```
      .hbox {
      -fx-border-color: blue;
      -fx-border-width: 2px;
      -fx-border-radius: 5px;
      -fx-border-insets: 5px;
      -fx-padding: 10px;
      -fx-spacing: 5px;
      -fx-background-color: lightgray;
      -fx-background-insets: 5px;
      }
      .button {
      -fx-text-fill: blue;
      }
      ```

      - 使用上面的 style sheet中的 selector

        ```
        import javafx.application.Application;
        import javafx.application.Platform;
        import javafx.scene.Scene;
        import javafx.scene.control.Button;
        import javafx.scene.control.Label;
        import javafx.scene.control.TextField;
        import javafx.scene.layout.HBox;
        import javafx.stage.Stage;
        public class StyleClassTest extends Application {
        public static void main(String[] args) {
                Application.launch(args);
        ```

```
    }
    @Override
    public void start(Stage stage) {
        Label nameLbl = new Label("Name:");
        TextField nameTf = new TextField("");
        Button closeBtn = new Button("Close");
        closeBtn.setOnAction(e -> Platform.exit());
        HBox root = new HBox();
        root.getChildren().addAll(nameLbl, nameTf, closeBtn);
        // Set the styleClass for the HBox to "hbox"
        root.getStyleClass().add("hbox");
        Scene scene = new Scene(root);
        scene.getStylesheets().add("resources/css/styleclass.css");
        stage.setScene(scene);
        stage.setTitle("Using Style Class Selectors");
        stage.show();
    }
}
```

> The Button class adds a style class, which is named "button", to all its instances
>
> 因此按钮字体是蓝色的（使用了button选择器)

- wraning :JavaFX中大多数常用的控件都有一个默认的样式类名称。 如果需要，您可以添加更多样式类名称。 默认样式类名称是从JavaFX类名构建的。 JavaFX类名转换为小写字母，连字符插入两个单词的中间。 如果JavaFX类名只包含一个单词，则相应的默认样式类名称只需将其转换为小写即可创建。 例如，默认样式类名称是Button的按钮，label for 标签，用于超链接的超链接，用于TextField的文本字段，用于TextArea的文本区域，用于CheckBox的复选框
- wraning :JavaFX container classes, for example, Region, Pane, HBox, VBox, do not have a default style class name

- 知道节点的默认样式类名称以在样式表中使用它
  - 由类名来猜测
  - 使用在线JavaFX CSS参考指南查找名称
  - Write a small piece of code

```
Button btn = new Button();
ObservableList<String> list = btn.getStyleClass();
if (list.isEmpty()) {
    System.out.println("No default style class name");
} else {
for(String styleClassName : list) {
    System.out.println(styleClassName);
}
}
```

13.2 Class Selector for the root Node

- The root node of a scene is assigned a style class named "root"
    - root node默认匹配style class 为 root selector

13.3 Using ID Selectors

- 选择器语法

```
Button b1 = new Button("Close");
b1.setId("closeBtn");

#closeButton{
    -fx-text-fill:red;
}
```

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
public class IDSelectorTest extends Application {
public static void main(String[] args) {
        Application.launch(args);
    }
    @Override
    public void start(Stage stage) {
        Button openBtn = new Button("Open");
        Button saveBtn = new Button("Save");
        Button closeBtn = new Button("Close");
        closeBtn.setId("closeButton");
        HBox root = new HBox();
        root.getChildren().addAll(openBtn, saveBtn, closeBtn);
        Scene scene = new Scene(root);
        scene.getStylesheets().add("resources/css/idselector.css");
        stage.setScene(scene);
        stage.setTitle("Using ID selectors");
        stage.show();
    }
}
```

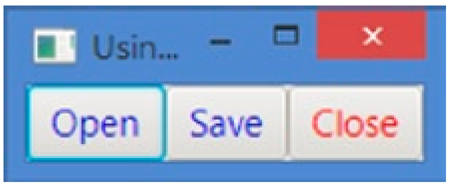


Figure 8-8. *Buttons using class and ID selectors*

13.4 Combining ID and Class Selectors

```
#closeButton.button{
    -fx-text-fill:red;
}
.button#closeButton{
    -fx-text-fill:red;
}
```

> 两者一样（顺序无关）
>
> 寻找 id是closeButton的button

13.5 The Universal Selector

```
*{
    -fx-text-fill:blue;
}
```

> 匹配所有
>
> 而
>
> *.button
>
> .button
>
> 两者是一样的

13.6 Grouping Multiple Selectors

```
.button {
    -fx-text-fill: blue;
}
.label {
    -fx-text-fill: blue;
}

// 上面两者结合
.button, .label {
    -fx-text-fill: blue;
}
```

13.7 Descendant Selectors

> 当您想要对JavaFX控件的某些部分进行样式化时，派生选择器会派上用场。 JavaFX中的许多控件由子节点组成，这些子节点是JavaFX节点

```
.hhox .button{
    -fx-text-fill :blue;
}

.check-box .text{
    -fx-fill:blue;
}

.check-box .box{
    -fx-border-color: black;
    -fx-border-width : 1px;
    -fx-border-style :dotted;
}
```

> 例如 CheckBox 包含 LabeldText子节点等

13.8 Child Selectors

> 注意区别，这边是孩子选择器
>
> 上面的 descendant是节点组成部分的部分选择

```
.hbox > .button {
    -fx-text-fill: blue;
}
```

13.9 State-Based Selectors

> 伪类选择器
>
> 取决于当前的state

```
.button:focused{  // button，并且处于 foucused

}
#openBtn:hover{   // id为openBtn的节点，并且鼠标处于悬浮（hover）于节点上方

}
```

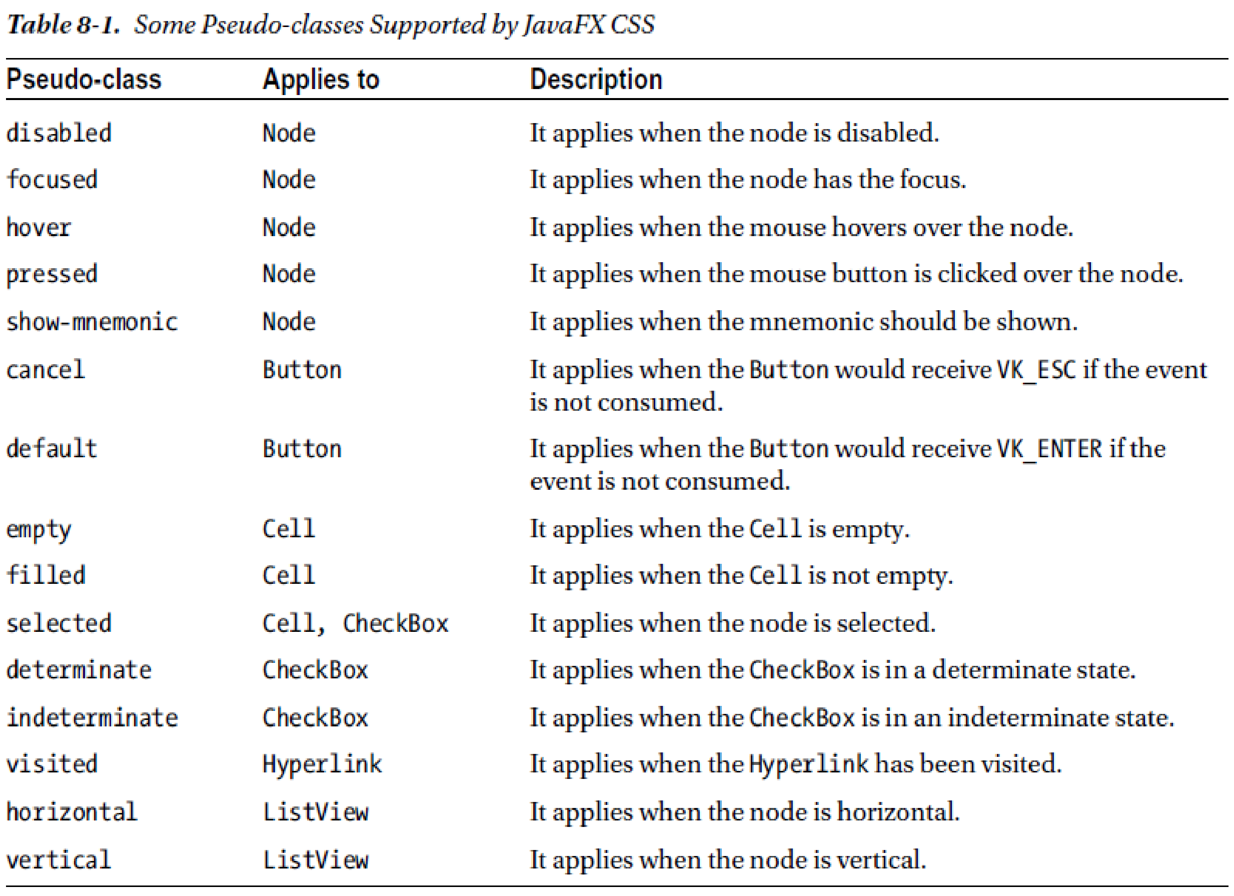


Table 8-1. Some Pseudo-classes Supported by JavaFX CSS

Pseudo-class	Applies to	Description
disabled	Node	It applies when the node is disabled.
focused	Node	It applies when the node has the focus.
hover	Node	It applies when the mouse hovers over the node.
pressed	Node	It applies when the mouse button is clicked over the node.
show-mnemonic	Node	It applies when the mnemonic should be shown.
cancel	Button	It applies when the Button would receive VK_ESC if the event is not consumed.
default	Button	It applies when the Button would receive VK_ENTER if the event is not consumed.
empty	Cell	It applies when the Cell is empty.
filled	Cell	It applies when the Cell is not empty.
selected	Cell, CheckBox	It applies when the node is selected.
determinate	CheckBox	It applies when the CheckBox is in a determinate state.
indeterminate	CheckBox	It applies when the CheckBox is in an indeterminate state.
visited	Hyperlink	It applies when the Hyperlink has been visited.
horizontal	ListView	It applies when the node is horizontal.
vertical	ListView	It applies when the node is vertical.

13.10 Using JavaFX Class Names as Selectors

- 允许但不推荐将JavaFX类名称用作样式中的类型选择器

```
HBox {
    -fx-border-color: blue;
    -fx-border-width: 2px;
    -fx-border-insets: 10px;
    -fx-padding: 10px;
}
Button {
    -fx-text-fill: blue;
}
```

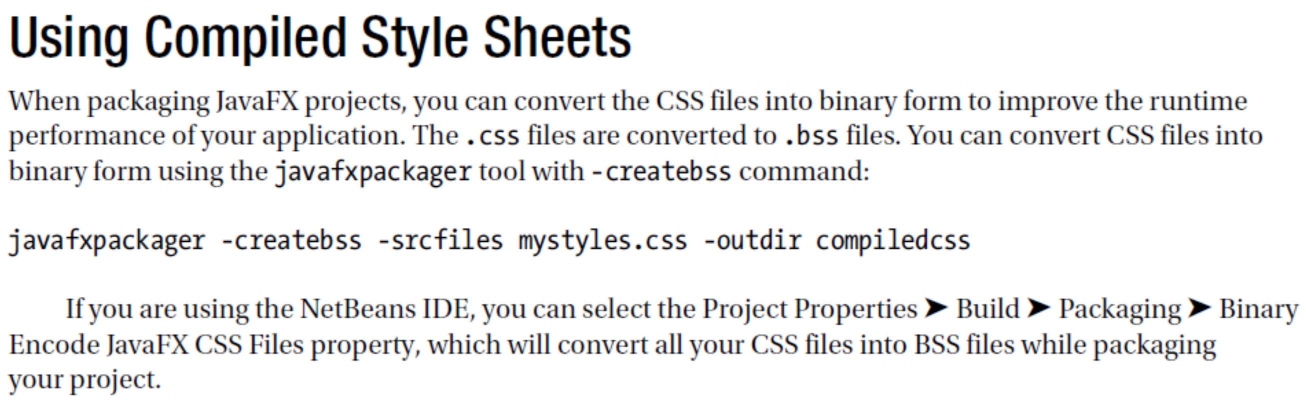
14. Looking Up Nodes in a Scene Graph

- lookup(String selector) method
    - class : Scene or Node
    - 返回找到的第一个节点的引用,没有则 null
    - The methods in two classes work a little differently

- 对于Scene，在整个scene graph中寻找
- 对于Node，在调用节点和子节点中寻找
  - lookupAll 则返回一个Set of all Nodes that are matched

```java
Button b1 = new Button("Close");
b1.setId("closeBtn");
VBox root = new VBox();
root.setId("myvbox");
root.getChildren().addAll(b1);
Scene scene = new Scene(root, 200, 300);
...
Node n1 = scene.lookup("#closeBtn"); // n1 is the reference of b1
Node n2 = root.lookup("#closeBtn"); // n2 is the reference of b1
Node n3 = b1.lookup("#closeBtn"); // n3 is the reference of b1
Node n4 = root.lookup("#myvbox"); // n4 is the reference of root
Node n5 = b1.lookup("#myvbox"); // n5 is null
Set<Node> s = root.lookupAll("#closeBtn"); // s contains the reference of b1
```

15. Using Compiled Style Sheets



Using Compiled Style Sheets

When packaging JavaFX projects, you can convert the CSS files into binary form to improve the runtime performance of your application. The `.css` files are converted to `.bss` files. You can convert CSS files into binary form using the `javafxpackager` tool with `-createbss` command:

```
javafxpackager -createbss -srcfiles mystyles.css -outdir compiledcss
```

If you are using the NetBeans IDE, you can select the Project Properties ➤ Build ➤ Packaging ➤ Binary Encode JavaFX CSS Files property, which will convert all your CSS files into BSS files while packaging your project.