JavaFX offer a great number of GUI Controls to use in applications. As any GUI toolkit must do...

Even if more than 20 different controls are available, there are still some missing. Most notably the dialogues.

By each version of JavaFX, more and more controls are added. With version 8 (in Java 8) most areas will have been covered, but there are already enough controls to get started.

We will have a look at many of the different controls in JavaFX. The purpose is to have an extra look at interesting parts of the control. Not all controls will be covered. We will concentrate on the more interesting and useful ones. There have also been some additions to the number of controls in JavaFX 2.1 and 2.2 that are not covered in the book. You are encouraged to play around with the components as well.
Label

- A Label is used to display text in a GUI, often in association to a text field or similar.
- In JavaFX a Label can be used to display text, image or both.
  - With the ability to adjust where to put the image in relation to the text.
- As a Label is a Node, it is also possible to add effects such as drop shadow to it.
  - The example will also show how to set a font for the text.
    - Any font available on the system is possible to use, as well as any point size.

The Code

```java
public void start(Stage primaryStage) {
    AnchorPane layout = new AnchorPane();
    Label theLabel = new Label("Star Wars Encyclopedia");
    theLabel.setFont(new Font("URWClassicoT", 42));
    Image ls = new Image(getClass().getResourceAsStream("lightsaber.jpg"));
    theLabel.setGraphic(new ImageView(ls));
    theLabel.setGraphicTextGap(10.0);
    theLabel.setContentDisplay(ContentDisplay.RIGHT);
    DropShadow ds = new DropShadow();
    ds.setOffsetX(5.0);
    ds.setOffsetY(5.0);
    theLabel.setEffect(ds);
    Label theIntro = new Label();
    theIntro.setText("Episode IV, A NEW HOPE It is a period of civil war. Rebel spaceships, striking from all directions, have now attacked the Home planet.
    
    theIntro.setWrapText(true);
    theIntro.setPrefSize(200.0, 200.0);
    theIntro.setRotate(-45.0);
    AnchorPane.setTopAnchor(theLabel, 5.0);
    AnchorPane.setLeftAnchor(theLabel, 5.0);
    AnchorPane.setBottomAnchor(theIntro, 20.0);
    AnchorPane.setLeftAnchor(theIntro, 50.0);
    layout.getChildren().addAll(theLabel, theIntro);
}
```

In graphics

```
Button

- Just as Label, the Button class inherits from Labeled and therefore many of the methods are the same.
  - For example how to add images and how to position them.
  - Setting the font is also possible, but better left to CSS as all buttons should have the same style.
- The example also shows how to use the constructor for adding an image directly.
- In this example, we also chain visual effects (a drop shadow and a reflection) and use mouse events to change the effect for a button.
```

```java
Button
```

```jav```
Radio and Check buttons with toggles

The RadioButton and CheckBox controls are quite similar.
- The first allows for one active choice and the other several active choices.
- In most cases both of them are put inside of a ToggleGroup to group the choices together.
- There are several ways of identifying the selected choice, but the easiest is to add UserData to each control choice.
- This can be used in the listener for the action of selecting an item.
- The event handler looks rather complicated but with the help of NetBeans, it is fairly easy to construct it.

The code, part 1

```java
BorderPane root = new BorderPane();
VBox radios = new VBox();
radios.setPadding(new Insets(5, 5, 5, 5));
radios.setSpacing(5.0);
radios.getChildren().addAll(radioBwing, radioCruiser, radioSSD);
```

The code, part 2

```java
AnchorPane layout = new AnchorPane();
final DropShadow ds = new DropShadow();
ds.setOffsetX(5.0); ds.setOffsetY(5.0);
```

Radio and Check buttons with toggles

◮ The RadioButton and CheckBox controls are quite similar.
◮ The first allows for one active choice and the other several active choices.
◮ In most cases both of them are put inside of a ToggleGroup to group the choices together.
◮ There are several ways of identifying the selected choice, but the easiest is to add UserData to each control choice.
◮ This can be used in the listener for the action of selecting an item.
◮ The event handler looks rather complicated but with the help of NetBeans, it is fairly easy to construct it.

The code, part 1

```java
BorderPane root = new BorderPane();
final Image ssd = new Image(getClass().getResourceAsStream("ssd.png"));
radioSSD.setUserData("ssd");
radioSSD.setToggleGroup(group);
radioSSD.setUserData("ssd");
radioSSD.setOnCheckedChangeListener((ov, old, new_toggle) -> {
    if (group.getSelectedToggle ()==null) {
        radioSSD.setImage(ssd);
    } else if (group.getSelectedToggle ()==radioSSD) {
        radioSSD.setImage(ssd);
    } else {
        radioSSD.setImage(null);
    }
});
```

The code, part 2

```java
final Image bwing = new Image(getClass().getResourceAsStream("bwing.png"));
final Image cruiser = new Image(getClass().getResourceAsStream("cruiser.png"));
final Image exit = new Image(getClass().getResourceAsStream("exit.png"));
final Image ok = new Image(getClass().getResourceAsStream("ok.png"));
```

The code, part 1

```java
public void handle(MouseEvent e) {
    AnchorPane.setRightAnchor(okButton, 15.0); AnchorPane.setBottomAnchor (okButton, 15.0);
    okButton.setEffect(ds);
    okButton.setUserData("Ok");
    okButton.setContentDisplay(ContentDisplay.TOP);
    okButton.addEventHandler (MouseEvent.MOUSE_EXITED,()
        { return;
    });
    okButton.addEventHandler (MouseEvent.MOUSE_ENTERED,(event)
        { return;
    });
}
```
In graphics

GUI Controls School of Computer Science, Physics and Mathematics
JavaFX Essentials 13(69)

ChoiceBox

◮ A ChoiceBox is similar in function to a drop box.
  ◮ A selection is made from a list of elements that are shown when clicking the box.
  ◮ From version 2.1 there is also the more advanced ComboBox which provides the same functionality.
  ◮ Simply put, use ChoiceBox for few items and ComboBox for many items.
  ◮ Also, use ComboBox if the added features are needed.
  ◮ In a ChoiceBox, the currently selected item is ticked.
  ◮ There are two ways of adding items to a ChoiceBox.
    ◮ Through `getItems.addAll()` or via `setItems(FXCollection)`.

Selecting a choice

◮ The method `getSelectionModel()` returns the selected item.
  ◮ Most often, the index number is wanted, therefore the method `selectedIndexProperty()` can be used.
    ◮ Add a listener to it in order for a change to be recognised by Java.
    ◮ To be able to use the index, the listener must be for the Number type.
    ◮ The `changed` method holds the action to be performed.
      ◮ It has the previous and the newly selected value index as input.
    ◮ There are, of course, other ways as well.

The code, part 1

```java
VBox layout = new VBox();
layout.setPadding(new Insets(5, 5, 5, 5));
ChoiceBox pinkFloyd = new ChoiceBox();
VBox imageBox = new VBox();
imageBox.setPadding(new Insets(25, 5, 5, 5));
Image theAlbum = new Image(getClass().getResourceAsStream("pf.jpg"));
final ImageView showAlbum = new ImageView(theAlbum);
showAlbum.setFitWidth(100.0);
showAlbum.setPreserveRatio (true);
Reflection refl = new Reflection();
refl.setFraction(0.8);
showAlbum.setEffect(refl);
imageBox.getChildren().add(showAlbum);
layout.getChildren().addAll(pinkFloyd, imageBox);
```
The code, part 2

```java
pinkFloyd.getSelectionModel().selectedIndexProperty().addListener(
    new ChangeListener <Number>(){
        public void changed(ObservableValue ov, Number oldValue, Number newValue){
            if (newValue==0)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("piper.jpg")));
            else if (newValue==1)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("saucer.jpg")));
            else if (newValue==2)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("more.jpg")));
            // some statements omitted
            else if (newValue==10)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("wall.jpg")));
            else if (newValue==11)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("final.jpg")));
            else if (newValue==12)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("mlor.jpg")));
            else if (newValue==13)
                showAlbum.setImage(new Image(getClass().getResourceAsStream("divbell.jpg")));
        }
    });

Scene scene = new Scene(layout, 300, 250);
primaryStage.setTitle("Pink Floyd");
primaryStage.setScene(scene);
primaryStage.show();
```

In graphics

```java
ScrollBar scroller = new ScrollBar();
scroller.setOrientation(Orientation.HORIZONTAL);
scroller.setMin(0);
scroller.setMax(645);
AnchorPane root = new AnchorPane();
AnchorPane.setBottomAnchor (scroller, 2.0);
AnchorPane.setLeftAnchor(scroller, 2.0);
AnchorPane.setRightAnchor(scroller, 2.0);
Image mars = new Image(getClass().getResourceAsStream("mars.jpg"));
final ImageView showMars = new ImageView(mars);
Rectangle2D view = new Rectangle2D(0, 0, 300, 350);
showMars.setViewport(view);
AnchorPane.setTopAnchor(showMars, 2.0);
AnchorPane.setLeftAnchor(showMars, 2.0);
scroller.valueProperty().addListener(
    new ChangeListener<Number>() {
        public void changed(ObservableValue<?> ov,
            Number old_val, Number new_val) {
            showMars.setViewport(
                new Rectangle2D((Double)new_val, 0, 300, 350));
        }
    });
root.getChildren().addAll(showMars, scroller);
Scene scene = new Scene(root, 300, 350);
primaryStage.setTitle("Hello Another World!");
primaryStage.setScene(scene);
primaryStage.show();
```
In graphics

List View

- In the ListView a number of items are displayed in a list.
- It allows for single or multiple selection as well as automatic scroll bars.
- There are a number of methods to listen to:
  - `getSelectionModel().getSelectedIndex()` — returns the selected index
  - `getSelectionModel().getSelectedItem()` — returns the selected item
  - `getFocusedModel().getFocusedIndex()` — returns the index of focused item
  - `getFocusedModel().getFocusedItem()` — returns the currently focused item

The code, part 1

```java
HBox layout = new HBox();
ListView<String> jediList = new ListView<>();
jediList.setPrefWidth(150);
ObservableList<String> theJedi = FXCollections.observableArrayList(
        "Yoda",
        "Mace Windu",
        "Plo Koon",
        "Ki-Adi-Mundi",
        "Shaak Ti",
        "Even Piell",
        "Oppo Rancisis",
        "Saesee Tinn",
        "Coleman Trebor",
        "Eeth Koth",
        "Adi Gallia",
        "Depa Billaba");
jediList.setItems(theJedi);

Image jediImage = new Image(getClass().getResourceAsStream("jediorder.png"));
final ImageView showJedi = new ImageView(jediImage);
showJedi.setFitWidth(200);
showJedi.setPreserveRatio(true);
final ArrayList<Image> manyJedi = new ArrayList<>();
manyJedi.add(new Image(getClass().getResourceAsStream("yoda.png")));
manyJedi.add(new Image(getClass().getResourceAsStream("macewindu.png")));
// lines omitted
manyJedi.add(new Image(getClass().getResourceAsStream("adigallia.png")));
manyJedi.add(new Image(getClass().getResourceAsStream("Depa.jpg")));
layout.getChildren().addAll(jediList, showJedi);
```

The code, part 2

```java
jediList.getSelectionModel().selectedIndexProperty().addListener(
        new ChangeListener<Number>() {
            public void changed(ObservableValue<? extends Number> ov, Number old_val, Number new_val) {
                showJedi.setImage(manyJedi.get((Integer)new_val));
            }
        });

Scene scene = new Scene(layout, 330, 250);
primaryStage.setTitle("Hello Jedi!");
primaryStage.setScene(scene);
primaryStage.show();
```
More on ListView

- Every item space is called a **cell**.
- It is also possible to add editing using **cell factories**.
  - Using for example a combo box or text field.
- This means that the space that occupies a cell, another component will be placed.
- A list of elements (not visual) needs to be set to this component.
- It is then possible to get the cell component by double clicking on the item.
  - In our example a menu of possible names will appear.

```java
final ObservableList jedi = FXCollections.observableArrayList();
final ObservableList data = FXCollections.observableArrayList();
final ListView jediCouncil = new ListView(data);
jediCouncil.setEditable(true);
for (int i = 0; i < 12; i++)
    data.add("vacant");
jediCouncil.setCellFactory(ComboBoxListCell.forListView(jedi));
StackPane root = new StackPane();
root.getChildren().add(jediCouncil);
Scene scene = new Scene(root, 300, 250);
primaryStage.setTitle("New Jedi Council");
primaryStage.setScene(scene);
primaryStage.show();
```
TreeView

> Constructing a tree view is a two part process.
  > First create a root element with children using `TreeItem`
  > Second create the view that shows the tree using `TreeView`
> The items themselves can contain any type of data – for example text and/or images.
> In the example, we create one root and several sub trees (all of them `TreeItem`).
> This is attached to the view which is set to be expanded with `setExpanded(true)`.
> A listener is attached to the selected item property to show the content in a label.

The code, part 1

```java
VBox layout = new VBox();
layout.setPadding(new Insets(5, 5, 5, 5));
layout.setSpacing(5);

TreeItem<String> starWarsRoot = new TreeItem<>
("Star Wars Movies",
new ImageView(new Image(getClass().getResourceAsStream("R2-D2.png"))));

starWarsRoot.setExpanded(true);

TreeItem<String> menaceRoot = new TreeItem<>
("The Phantom Menace",
new ImageView
(new Image(getClass().getResourceAsStream("Naboo Starfighter.png"))));

menaceRoot.getChildren().add(new TreeItem<>
("Anakin Skywalker"));
menaceRoot.getChildren().add(new TreeItem<>
("Qui-Gon Jinn"));
menaceRoot.getChildren().add(new TreeItem<>
("Obi-Wan Kenobi"));
menaceRoot.getChildren().add(new TreeItem<>
("Jar-Jar Binks"));
menaceRoot.getChildren().add(new TreeItem<>
("Queen Amidala"));

TreeItem<String> attackRoot = new TreeItem<>
("Attack of the Clones",
new ImageView(new Image(getClass().getResourceAsStream("clone-1.png"))));

attackRoot.getChildren().add(new TreeItem<>
("Jango Fett"));
attackRoot.getChildren().add(new TreeItem<>
("Cliegg Lars"));
attackRoot.getChildren().add(new TreeItem<>
("Count Dooku"));
attackRoot.getChildren().add(new TreeItem<>
("Poggle the Lesser"));
attackRoot.getChildren().add(new TreeItem<>
("Onaconda Farr"));

// Four more sub trees are created here.

starWarsRoot.getChildren().addAll(menaceRoot, attackRoot, revengeRoot,
hopeRoot, empireRoot, returnRoot);

TreeView<String> starWars = new TreeView<String>(starWarsRoot);

final Label selectedCharacter = new Label();

starWars.getSelectionModel().selectedItemProperty().addListener(
new ChangeListener<TreeItem<String>>() {
  public void changed(ObservableValue<? extends TreeItem<String>> ov,
                      TreeItem<String> old_val, TreeItem<String> new_val) {
    selectedCharacter.setText(new_val.getValue());
  }
});

layout.getChildren().addAll(starWars, selectedCharacter);

Scene scene = new Scene(layout, 300, 350);

primaryStage.setTitle("Star Wars Movies");

primaryStage.setScene(scene);

primaryStage.show();
```

The code, part 2

```java
starWarsRoot.getChildren().addAll(menaceRoot, attackRoot, revengeRoot,
hopeRoot, empireRoot, returnRoot);

TreeView<String> starWars = new TreeView<String>(starWarsRoot);

final Label selectedCharacter = new Label();

starWars.getSelectionModel().selectedItemProperty().addListener(
new ChangeListener<TreeItem<String>>() {
  public void changed(ObservableValue<? extends TreeItem<String>> ov,
                      TreeItem<String> old_val, TreeItem<String> new_val) {
    selectedCharacter.setText(new_val.getValue());
  }
});

layout.getChildren().addAll(starWars, selectedCharacter);

Scene scene = new Scene(layout, 300, 350);

primaryStage.setTitle("Star Wars Movies");

primaryStage.setScene(scene);

primaryStage.show();
```

In graphics
Sliders

- A slider is a graphical range indicator.
- It has a minimum value and a maximum value.
- It is possible to define the step per click or when clicking the bar itself.
- Graphically, it is possible to show tick marks and values on the slider.
- To listen to the changes, use a listener on the valueProperty.

Example

```java
VBox root = new VBox();
root.setPadding(new Insets(5, 5, 5, 5));
root.setSpacing(5);
Image theFish = new Image(getClass().getResourceAsStream("nemo.png"));
final ImageView fishShow = new ImageView(theFish);
final Rectangle2D rect = new Rectangle2D(0, 0, 994, 686);
fishShow.setViewport(rect);
fishShow.setFitWidth(290);
fishShow.setPreserveRatio(true);
fishShow.setVisible(false);
Slider zoom = new Slider();
zoom.setMin(0);
zoom.setMax(1.0);
zoom.setMajorTickUnit(0.5);
zoom.setShowTickLabels(true);
zoom.setShowTickMarks(true);
zoom.valueProperty().addListener(new ChangeListener<Number>() {
    @Override
    public void changed(ObservableValue<? extends Number> ov, Number old_val, Number new_val) {
        fishShow.setVisible(true);
        fishShow.setScaleX((Double)new_val);
        fishShow.setScaleY((Double)new_val);
    }
});
```

In graphics

![Progress indicators](image.png)

Progress indicators

- There are two different types of progress indicators in JavaFX:
  - ProgressBar – a line to indicate the progress.
  - ProgressIndicator – a circle diagram to show the progress.
- Both of them take a value from 0.0 to 1.0 as the percentage done.
  - A negative value will place the progress in "indeterminate" status.
- The value is changed using the setProgress() method.
The code

```java
VBox layout = new VBox();
layout.setPadding(new Insets(5, 5, 5, 5));
layout.setSpacing(5);
final ListView<String> seven = new ListView<>();
seven.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
seven.setPrefHeight(200);
seven.setItems(dwarfs);
final ProgressBar theProgress = new ProgressBar(0.0);
theProgress.setPrefWidth(295);
final ProgressIndicator theIndicator = new ProgressIndicator(0.0);
theIndicator.setPrefSize(100, 100);
seven.getSelectionModel().selectedIndexProperty().addListener((ChangeListener<Number>)
    public void changed(ObservableValue<? extends Number> ov, Number old_val, Number new_val) {
        int numItems = seven.getSelectionModel().getSelectedItems().size();
        if (numItems <= 7) {
            theProgress.setProgress(numItems / 7.0);
            theIndicator.setProgress(numItems / 7.0);
        }
    });
layout.getChildren().addAll(seven, theProgress, theIndicator);
Scene scene = new Scene(layout, 300, 300);
```

In graphics

![TitledPane and Accordion](image)

**TitledPane and Accordion**

- A `TitledPane` is a pane that can be maximised and minimised.
  - It also has a title...
  - It can be used by itself, but is most often used together with other controls, like the Accordion.
  - With that control a group of titled panes can be added.
  - It is possible to activate one pane by selecting it, while the others stay minimised.

The code

```java
TitledPane thePane = new TitledPane();
thePane.setText("Saesee Tiin");
ImageView theImage = new ImageView(new Image(getClass().getResourceAsStream("saeseetiin.png")));
theImage.setFitWidth(300);
theImage.setPreserveRatio(true);
thePane.setContent(theImage);
StackPane root = new StackPane();
root.getChildren().add(thePane);
Scene scene = new Scene(root, 300, 545);
primaryStage.setTitle("Hello Saesee Tiin!");
primaryStage.setScene(scene);
primaryStage.show();
```
Code for an accordion

TitledPane tiinPane = new TitledPane();
tiinPane.setText("Saesee Tiin");
ImageView theImage = new ImageView(new Image(getClass().getResourceAsStream("saeseetiin.png")));
theImage.setFitHeight(300);
theImage.setPreserveRatio(true);
tiinPane.setContent(theImage);

TitledPane macePane = new TitledPane();
macePane.setText("Mace Windu");
ImageView imageMace = new ImageView(new Image(getClass().getResourceAsStream("macewindu.png")));
imageMace.setFitHeight(300);
imageMace.setPreserveRatio(true);
macePane.setContent(imageMace);

/* Several panes removed */

Accordion dragspel = new Accordion();
dragspel.getPanes().addAll(tiinPane, macePane, yodaPane, ploPane, tiPane, adiPane, piellPane);
dragspel.setExpandedPane(tiinPane);

StackPane root = new StackPane();
root.getChildren().add(dragspel);

Scene scene = new Scene(root, 300, 500);
primaryStage.setTitle("Hello Jedi Council!");
primaryStage.setScene(scene);
primaryStage.show();

Creating menus

In many of the programs used today, a menu is available for selection functions.

- Normally it is positioned at the top of the application.
  - Most applications only have just one menu.

In JavaFX, creating a menu is a three step process:

1. Create the MenuBar.
2. Create menus on the menu bar using Menu.
3. Create individual menu items using MenuItem.

- The menu item can also be of RadioButtonItem or CheckMenuItem type.
- It is also possible to create sub menus or context menus.
More on menus

- Each menu item can have an image associated with it.
- It is also possible to add an accelerator key for fast access.
- Sometimes a separator is needed between items, this is added using the `SeparatorMenuItem`.
- To add an action to a menu item, just implement the `setOnAction()` method.
- It needs an `EventHandler` and has the `handle` method.

```java
final Label theText = new Label();
theText.setWrapText(true);
final ImageView theView = new ImageView();
Image theImage = new Image(getClass().getResourceAsStream("spyro.png"));
theView.setImage(theImage);

MenuBar menuBar = new Menubar();
Menu menuFile = new Menu("File");
Menu menuWater = new Menu("Water");
Menu menuTech = new Menu("Technology");
Menu menuEffects = new Menu("Effects");
menuBar.getMenus().addAll(menuFile, menuWater, menuTech, menuEffects);

MenuItem exitItem = new MenuItem("Exit", new ImageView(new Image(getClass().getResourceAsStream("application_exit.png"))));
exitItem.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent t){
        Platform.exit();
    }
});
exitItem.setAccelerator(KeyCombination.keyCombination("Ctrl+X"));

MenuItem thumpback = new MenuItem("ThumpBack");
thumpback.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent t){
        theView.setImage(new Image(getClass().getResourceAsStream("thumpback.png")));
    }
});

try {
    theText.setText(readFile("thumpback.txt"));
} catch (FileNotFoundException ex) {
    Logger.getLogger(JavaFX_L4_13_Menu.class).log(Level.SEVERE, null, ex);
}

final ToggleGroup radioEffects = new ToggleGroup();
RadioMenuItem noEffects = new RadioMenuItem("None");
noEffects.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e) {
        theView.setEffect(null);
    }
});
noEffects.setToggleGroup (radioEffects);

RadioMenuItem reflEffects = new RadioMenuItem("Reflection");
reflEffects.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e) {
        theView.setEffect(refl);
    }
});
reflEffects.setToggleGroup (radioEffects);

RadioMenuItem dropEffects = new RadioMenuItem("Dropshadow");
dropEffects.setOnAction(new EventHandler<ActionEvent>() {
    public void handle(ActionEvent e) {
        theView.setEffect(ds);
    }
});
dropEffects.setToggleGroup (radioEffects);
```

VBox root = new VBox();
root.setPadding(new Insets(0, 5, 5, 5));
root.setSpacing(5);
root.getChildren().addAll(menuBar, theView, theText);

Scene scene = new Scene(root, 665, 550);
primaryStage.setTitle("Characters from Skylanders");
primaryStage.setScene(scene);
primaryStage.show();
In graphics

Tabs and the web

◮ Our last example of the control tour will use three new controls:
  ◮ A TabPane with Tabs for switching between different functionality.
  ◮ An HTMLEditor for editing HTML code.
  ◮ A WebView using a WebEngine to show the result.
◮ To each Tab a content (any kind of Node) is set by using the `setContent()` method.
◮ The HTML editor is quite feature rich, but unfortunately it is not possible to change the visible parts of it – it is all or nothing.
◮ It is possible to do this using "illegal" CSS styling, but that might not work in the future.
◮ The web viewing capabilities are based on the WebKit web engine and is really impressive.

The code, part 1

```java
TabPane tabs = new TabPane();
Tab design = new Tab("Design");
design.setClosable(false);
Tab source = new Tab("Source");
source.setClosable(false);
Tab view = new Tab("View");
view.setClosable(false);
tabs.getTabs().addAll(design, source, view);
final HTMLEditor htmlEditor = new HTMLEditor();
design.setContent(htmlEditor);
final TextArea htmlSource = new TextArea();
htmlSource.setWrapText(true);
source.setContent(htmlSource);
final WebView browser = new WebView();
final WebEngine showWeb = browser.getEngine();
view.setContent(browser);
```

The code, part 2

```java
tabs.getSelectionModel().selectedIndexProperty().addListener(
    new ChangeListener<Number>() {
      public void changed(ObservableValue<? extends Number> ov, Number old_val, Number new_val) {
        if (new_val == 1) {
          htmlSource.setText(htmlEditor.getHtmlText());
        } else if (new_val == 2) {
          showWeb.loadContent(htmlEditor.getHtmlText());
        }
      }
    });
```

```java
Scene scene = new Scene(root, 600, 400);
primaryStage.setTitle("Ultimate Web Designer Deluxe -- Professional Edition");
primaryStage.setScene(scene);
primaryStage.show();
```
In graphics

More controls...?

Yes, there are many more controls available that we have not covered.

- ToolBar
- HyperLink
- ToolTip
- Pagination
- And several more...

This introduction, though, will have helped you to see the basic usage of most controls.

Learning more will not be a problem for you.

SCENE BUILDER

SceneBuilder

In many cases, the design of an application should not be left to the programmer.

To facilitate the separation of design and logic, Oracle has developed a designer tool called SceneBuilder.

The focus of this tool is to design the GUI and supply means for the programmer to hook up on to the design.

This is mainly done by:

- Giving each control an id
- Saving the GUI in FXML
- Load the FXML in JavaFX and access the ids using an adapter class

This part will show a walk through for doing all of this.
Getting started

- SceneBuilder is a separate download from javafx.com.
  - Currently, version 1.0 is available for Windows and MacOS X while the 1.1 beta is available for Linux as well.
- After downloading, just unpack or install the files.
- For extra convenience it is possible to add the path of the SceneBuilder to NetBeans.
  - Go to Tools → Options → Java → JavaFX tab.
- NetBeans will now automatically open FXML files in SceneBuilder.

Starting a new project

- Create a new project but choose the JavaFX FXML Application instead of the normal project type.
- This will create a skeleton FXML as well as a adapter (or controller) class for the communication.
- In the “Name and Location” step you need to give the FXML file a name in addition to the normal parts.
  - This will also create a controller class automatically.
- In our project there will now be three files:
  - The “main” application that starts the program, this will not be used further.
  - An FXML file.
  - A controller class which will be holding our logic.

The main file

- The only purpose of the main file is to load the FXML and display it in the scene.

```java
public void start(Stage stage) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("LegoGUI.fxml"));
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
}
```
- Nothing more is needed here and in fact this file will remain untouched for most parts.

The controller

- The controller class holds the logic for the application.
- In it, all the parts of the GUI are declared with the @FXML annotation.

```java
@FXML
private Label label;
```
- This shows the default parts, which we will replace shortly.
The FXML file

- The FXML file can be viewed "as-is" in NetBeans and it really is nothing but an XML file with JavaFX tags.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<javafx:AnchorPane id="AnchorPane" prefHeight="200" prefWidth="320" xmlns:fx="http://javafx.com/fxml"
>...
</javafx:AnchorPane>
```

- If we double click (or right click and choose "Open") it will open in SceneBuilder instead.

In SceneBuilder

Creating a new GUI

- Remove the parts in the GUI and create something like the following:

- In this case a border pane has been used with a VBox to the left with the buttons.
- The large area in the middle is a ImageView.

Adding id and action

- On the right side of the tool, there are a number of options in an accordion.
  
  - Depending on the selected control, but often Properties, Layout and Code.

- To make sure a control is attached to a layout border, use the Layout pane and the "cross".

- Under "Properties", give the buttons and the image view an fx:id.

- Under "Code" for the buttons, name an action method for each button.
  - They need to start with #.
Coding

- When all parts are added to the GUI it is time to create the code.
- In SceneBuilder, a skeleton code for the present GUI is generated.
- Go to View → Show Sample Controller Skeleton.
- Copy the code that is in the dialogue.
- Make certain that you only copy the relevant parts.
- Return to NetBeans and in the controller class, paste the code.
- Now, add code in the methods like previously.
- Repeat this every time a new GUI element has been added – but beware not to overwrite code!

Skeleton code in window

In graphics

Code after editing in NetBeans

In graphics
More to SceneBuilder?

- Yes, there is of course more to SceneBuilder!
- One important part is that it is possible to do the previous
  “the other way around”. If we add action methods in our code, they will appear in the
  drop down box in SceneBuilder.
- This means that SceneBuilder should be kept running for as
  long as any coding is done.
- The integration with NetBeans is today perhaps not as good
  as we would like, but that might change.
- More functionality will be added as time goes by.