User Interface Programming
Qt / QML

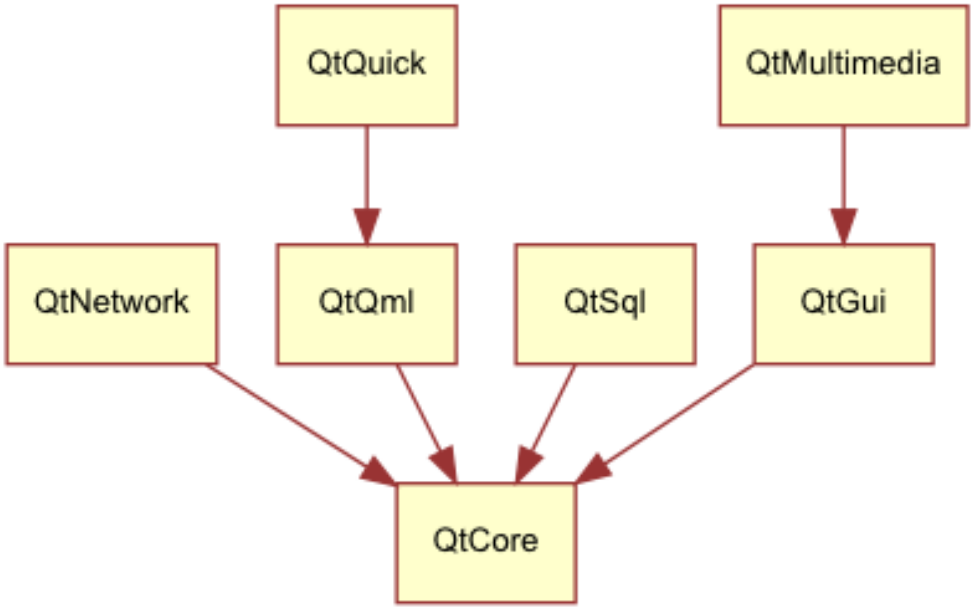
Jozef Mlích

Department of Computer Graphics and Multimedia
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
http://www.fit.vutbr.cz/~imlich
imlich@fit.vutbr.cz

23. 10. 2018

Outline

- QML
- Modules
- Syntax, Properties, Scripting
- Debugging
- Positioning
- Own components
- Controls
- PropertyAnimation/NumberAnimation/Behavior
- MVC
- C++ and QML



- Qt QML – Classes for QML and JavaScript languages.
- Qt Quick – declarative framework for building highly dynamic applications with custom user interfaces.
- Qt GUI – Base classes for graphical user interface (GUI) components. Includes OpenGL.

- qmlscene (qt4 - qmlviever)
- Compiled app

```
QQmlApplicationEngine engine;
engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
QObject *topLevel = engine.rootObjects().value(0);
QQuickWindow *window = qobject_cast<QQuickWindow *>(topLevel);
window->show();
```

QML Syntax

```
import QtQuick 2.5
// The root element is the Rectangle
Rectangle {
    // name this element root
    id: root
    // properties: <name>: <value>
    width: 120; height: 240
    // color property
    color: "#4A4A4A"
    // Declare a nested element (child of root)
    Image {
        id: triangle
        // reference the parent
        x: (parent.width - width)/2; y: 40
        source: 'assets/triangle_red.png'
    }
    // Another child of root
    Text {
        // un-named element
        // reference element by id
        y: triangle.y + triangle.height + 20
        // reference root element
        width: root.width
        color: 'white'
        horizontalAlignment: Text.AlignHCenter
        text: 'Triangle'
    }
}
```

Properties

```
import QtQuick 2.9
Text {
    // (1) identifier
    id: thisLabel
    // (2) set x- and y-position
    x: 24; y: 16
    // (3) bind height to 2 * width
    height: 2 * width
    // (4) custom property
    property int times: 24
    // (5) property alias
    property alias anotherTimes: thisLabel.times
    // (6) set text appended by value
    text: "Greetings " + times
    // (7) font is a grouped property
    font.family: "Ubuntu"
    font.pixelSize: 24
    // (8) KeyNavigation is an attached property
    KeyNavigation.tab: otherLabel
    // (9) signal handler for property changes
    onHeightChanged: console.log('height:', height)
    // focus is need to receive key events
    focus: true
    // change color based on focus value
    color: focus?"red":"black"
}
```

Scripting

```
import QtQuick 2.9
Text {
        id: label
        x: 24; y: 24
        // custom counter property for space presses
        property int spacePresses: 0
        text: "Space pressed: " + spacePresses + " times"
        // (1) handler for text changes
        onTextChanged: console.log("text changed to:", text)
        // need focus to receive key events
        focus: true
        // (2) handler with some JS
        Keys.onSpacePressed: {
            increment()
        }
        // clear the text on escape
        Keys.onEscapePressed: {
            label.text = ''
        }
        // (3) a JS function
        function increment() {
            spacePresses = spacePresses + 1
        }
}
```

Debugging

```
function f() {
    var x = 12
    console.assert(x == 12, "This will pass");
    console.assert(x > 12, "This will fail");
}
function f() {
    console.time("wholeFunction");
    console.time("firstPart");
    // first part
    console.timeEnd("firstPart");
    // second part
    console.timeEnd("wholeFunction");
}
function f() {
    console.log("Hello log");
    console.error("Hello error");
    console.count("f called");
}
function f() {
    console.profile();
//Call some function that needs to be profiled.
//Ensure that a client is attached before ending
//the profiling session.
    console.profileEnd();
}
```

```
Component.onCompleted: {
    console.log("Hello component");
}
```

- Item
- Rectangle
- Text
- TextInput
- Image
- MouseArea

Components

```
// Button.qml
import QtQuick 2.5
Rectangle {
    id: root
    // export button properties
    property alias text: label.text
    signal clicked
    width: 116; height: 26
    color: "lightsteelblue"
    border.color: "slategrey"
    Text {
        id: label
        anchors.centerIn: parent
        text: "Start"
    }
    MouseArea {
        anchors.fill: parent
        onClicked: {
            root.clicked()
        }
    }
}
```

```
// minimal API for a button
Button {
    text: "Click Me"
    onClicked: { // do something }
}
```

Positioning – Layout, Absolute, Anchors, ..

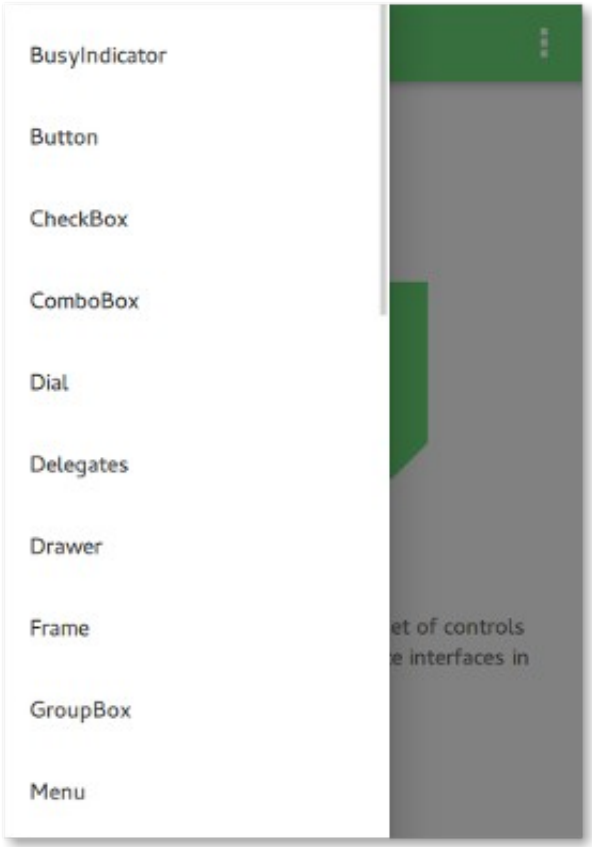
```
// column.qml
import QtQuick 2.5
DarkSquare {
    id: root
    width: 120
    height: 240
    Column {
        id: row
        anchors.centerIn: parent
        spacing: 8
        RedSquare { }
        GreenSquare { width: 96 }
        BlueSquare { }
    }
}
```

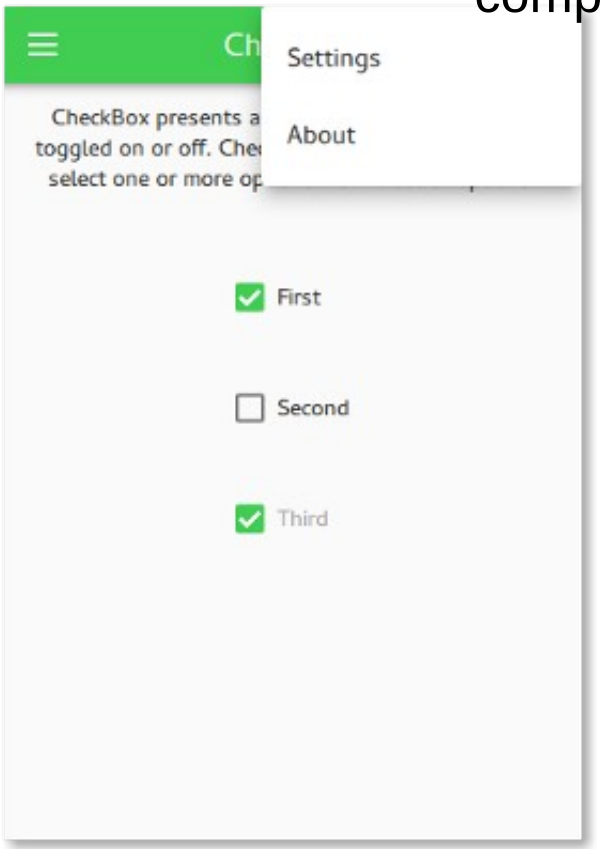
```
import QtQuick 2.9

Rectangle {
    width: 100
    height: 100
    Rectangle {
        id: first
        x: 10
        y: 10
        height: 10
        width: 10
        color: "red"
    }
    Rectangle {
        anchors.left: first.right;
        anchors.top: first.bottom;
        height: 10
        width: 10
        color: "blue"
    }
}
```

import QtQuick.Controls 2.4

- Qt Quick Controls 2 provides a set of controls that can be used to build complete interfaces in Qt Quick.



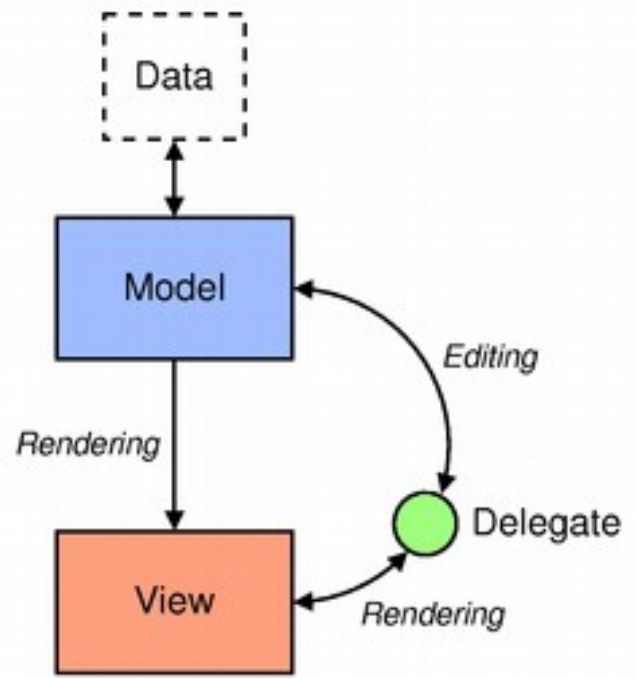


NumberAnimation, Behavior

```
    ClickableImageV2 {
        id: blueBox
        x: (root.width-width)/2; y: root.height-height
        source: "assets/box_blue.png"
        text: "behavior on property"
        Behavior on y {
            NumberAnimation { duration: 4000 }
        }

        onClicked: y = 40
        // random y on each click
//        onClicked: y = 40+Math.random()*(205-40)
    }
```

Design pattern MVC



- Model / View / Controller (Delegate)

```
// ContactsModel.qml
import QtQuick 2.0
import QtQuick.XmlListModel 2.0

ListModel {
     ListElement {
         name: "Bill Smith"
         number: "555 3264"
     }
     ListElement {
         name: "John Brown"
         number: "555 8426"
     }
     ListElement {
         name: "Sam Wise"
         number: "555 0473"
     }
}
```

```
import QtQuick 2.0
import QtQuick.XmlListModel 2.0

ListView {
     width: 180; height: 200
     model: ContactsModel {}
     delegate: Text {
         text: name + ": " + number
     }
}
```

```
class MyCppObject {
  Q_PROPERTY(QString foo ............)
  ....
};
qmlRegisterType<MyCppObject>("my.namespace", 1, 0, "MyQMLObject");


import my.namespace 1.0
MyQMLObject {
  foo: “string”;
}
```

References

- Qt documentation http://doc.qt.io/
- Juergen Bocklage-Ryannel and Johan Thelin: Qt 5 Cadaques https://qmlbook.github.io/
- https://wiki.qt.io/Books

Questions ?