Components

- It’s a good idea to recycle your code and create components for elements that can be reused.
- To create a component, place a tree of elements in a file with a `.qml` extension.
    - The tree is then available as a reusable component with the same name as the file but without the `.qml` extension.
- Example:

ButtonSimple.qml

```
import QtQuick 1.0

Image {
    id: button
    source: "images/button.png"

    Text {
        id: label
        text: "Push me!"
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

    MouseArea {
       anchors.fill: parent
       onClicked: {
           label.text = "Thanks!";
       }
   }
}
```

reusing-buttonsimple.qml

```
import QtQuick 1.0

Item {
    id: myItem
    width: 400
    height: 400

    ButtonSimple {
        id: myButton
        anchors.horizontalCenter: myItem.horizontalCenter
```

```
        anchors.bottom: myItem.bottom
        anchors.bottomMargin: 20
    }
}
```

- `ButtonSimple.qml` and `reusing-buttonsimple.qml` must be in the same directory.

Properties

- You can declare a `property` to expose a component's internal propery to the outside:

ButtonSimple2.qml

```
import QtQuick 1.0

Image {
    id: button
    source: "images/button.png"

    property string labelText

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

    MouseArea {
       anchors.fill: parent
       onClicked: {
           label.text = "Thanks!";
       }
   }
}
```

reusing-buttonsimple2.qml

```
import QtQuick 1.0

Item {
    id: myItem
```

```
    width: 400
    height: 400

    ButtonSimple2 {
        id: myButton
        labelText: "Hey you!"
        anchors.horizontalCenter: myItem.horizontalCenter
        anchors.bottom: myItem.bottom
        anchors.bottomMargin: 20
    }
}
```

Signals

- A `signal` declaration is similar to a `property` declaration except that it's used to let the outside world determine behavior.
- In the Button below, `buttonClicked` is declared as a signal, and that signal is emitted whenever the MouseArea is clicked.

Button.qml

```
import QtQuick 1.0

Image {
    id: button
    source: "images/button.png"

    property string labelText
    signal buttonClicked()

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

    MouseArea {
       anchors.fill: parent
       onClicked: {
           button.buttonClicked();
       }
   }
```

```
}
```

- The actual behavior is defined in the file where Button is used (as `onButtonClicked`):

resuing-button.qml

```
import QtQuick 1.0

Item {
    id: myItem
    width: 400
    height: 400

    Button {
        id: button
        labelText: "PRESS"
        anchors.horizontalCenter: myItem.horizontalCenter
        anchors.bottom: myItem.bottom
        anchors.bottomMargin: 20

        onButtonClicked:{
            button.labelText="Hooray!";
        }
    }
}
```

Resources

- images.zip
- All source code is subject to this copyright.