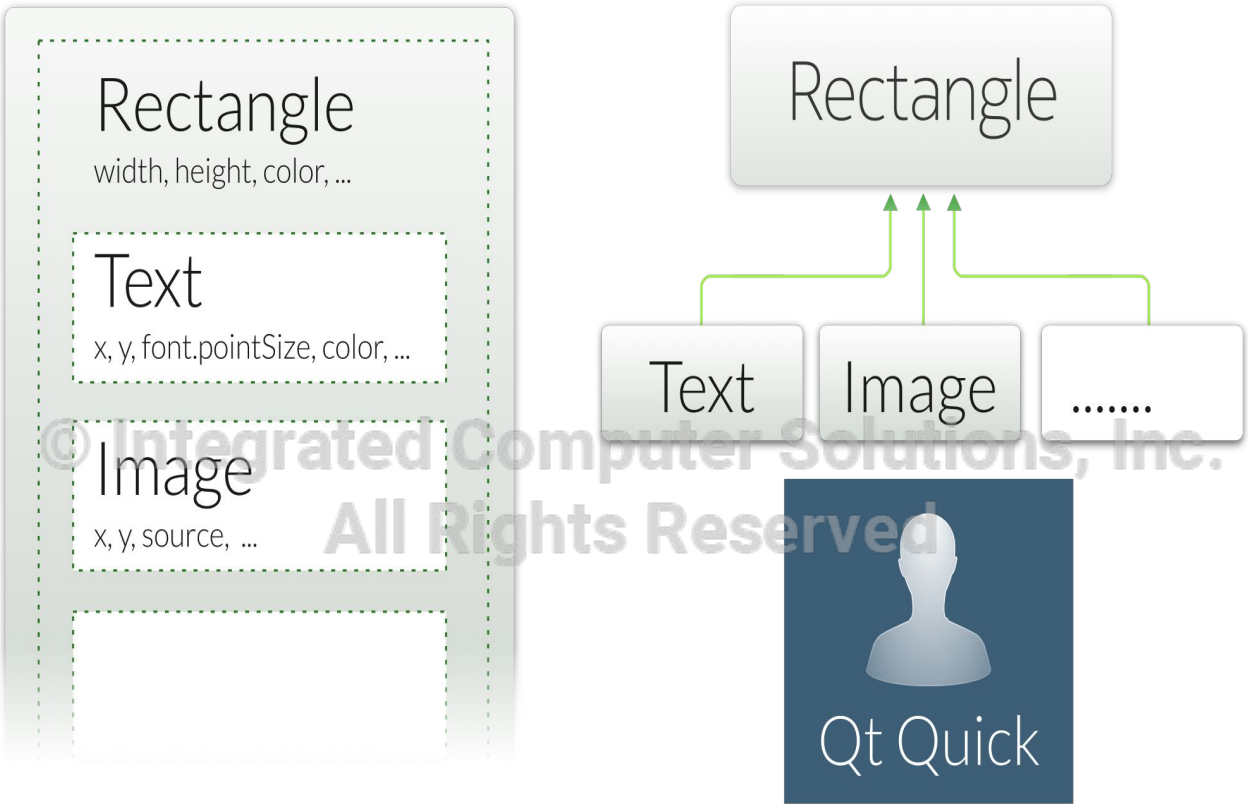
Qt For Beginners Training

Part 3

A Tree of QML Objects



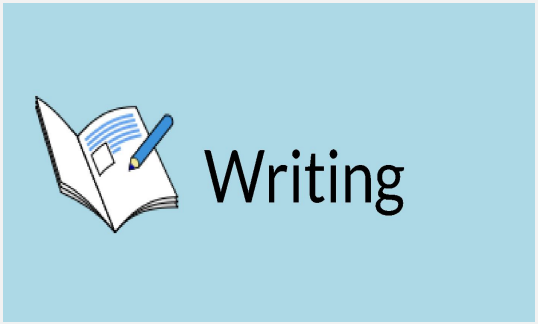
Qml Example

```
import QtQuick 2.15
Item {
    width: 300; height: 115
    Text {
        id: title
        x: 50; y: 25
        text: "Qt Quick"
        font { family: "Helvetica"; pointSize: parent.width * 0.1 }
    }

    Rectangle {
        x: title.x; y: title.y + title.height - height; height: 5
        width: title.width
        color: "green"
    }
}
```

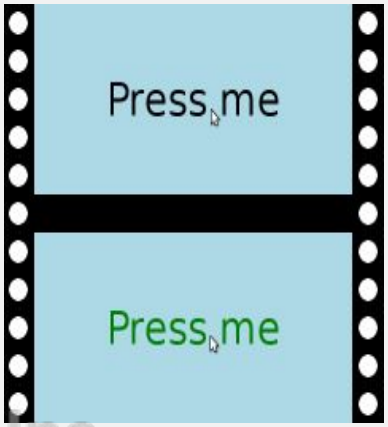
Using Anchors

```
Rectangle {
    width: 400; height: 200; color: "lightblue"
    Image {
        id: book; source: "../images/book.svg"
        anchors.left: parent.left
        anchors.leftMargin: parent.width / 16
        anchors.verticalCenter: parent.verticalCenter
    }
    Text {
        text: qsTr("Writing"); font.pixelSize: 32
        anchors.left: book.right; anchors.leftMargin: 32
        anchors.baseline: book.verticalCenter
    }
}
```



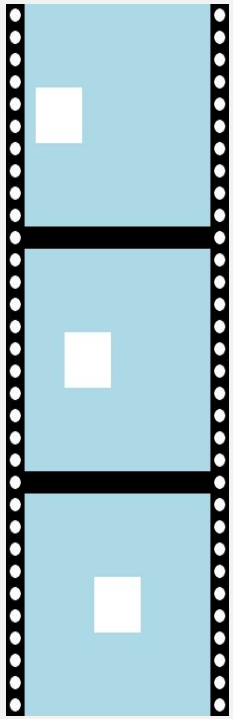
User Interaction

```
Rectangle {
    width: 400; height: 200; color: "lightblue"
    Text {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        text: qsTr("Press me"); font.pixelSize: 48
        MouseArea {
            anchors.fill: parent
            onPressed: parent.color = "green"
            onReleased: parent.color = "black"
        }
    }
}
```



Number Animations Revisited

```
Rectangle {
    width: 400; height: 400; color: "lightblue"
    Rectangle {
        id: rect
        x: 0; y: 150; width: 100; height: 100
    }
    NumberAnimation {
        target: rect
        properties: "x"
        from: 0; to: 150; duration: 1000
        running: true
    }
}
```

Defining a Custom Item

- Simple line edit
  - Based on undecorated **`TextInput`**
  - Stored in file **`TextInput.qml`**


```
Rectangle {
    border.color: "green"
    color: "white"
    radius: 4; smooth: true
    TextInput {
        anchors.fill: parent
        anchors.margins: 2
        text: qsTr("Enter text...")
        color: focus ? "black" : "gray"
        font.pixelSize: parent.height - 4
    }
}
```

Using a Custom Item

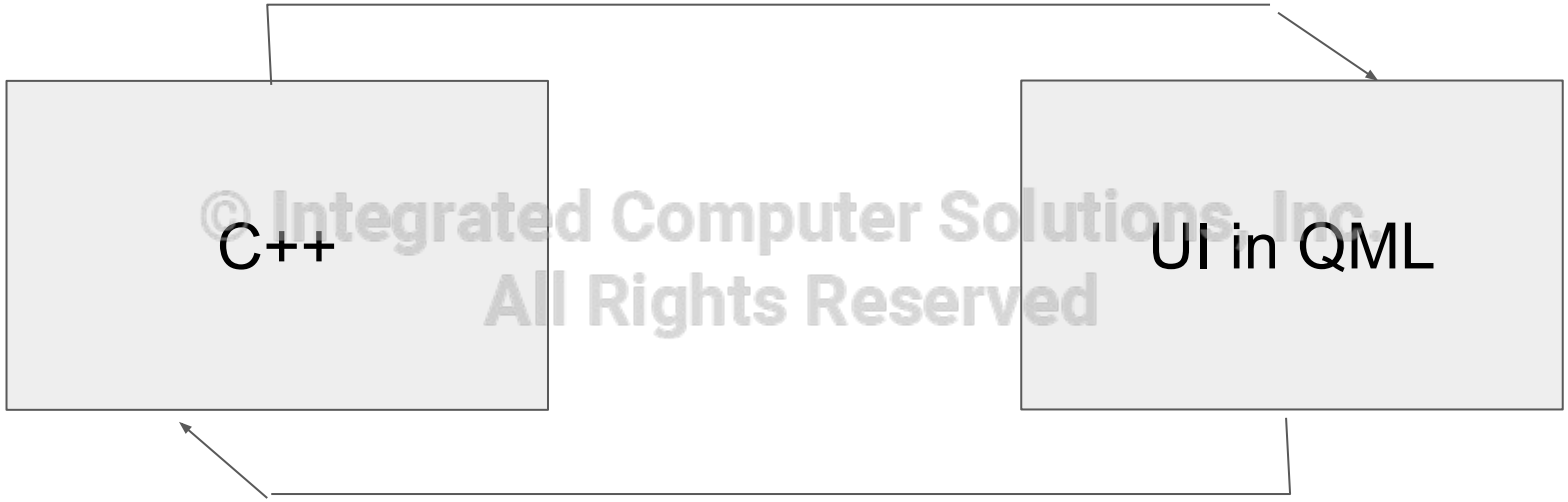
- **`LineEdit.qml`** is in the same directory
  - Item within the file automatically available as `LineEdit`

```
Rectangle {
    width: 400; height: 100; color: "lightblue"
    LineEdit {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        width: 300; height: 50
    }
}
```

Business Logic/UI Paradigm

-

- Signals
- Properties



- Q_INVOKABLE Function/ SLOT Calls

- C++ code “knows” nothing of the implementation details in QML

Exporting C++ Objects to QML

- C++ objects can be exported to QML

```
Class User : public QObject {
    Q_OBJECT
    Q_PROPERTY(QString name READ name WRITE setName NOTIFY nameChanged)
    Q_PROPERTY(int age READ age WRITE setAge NOTIFY ageChanged)
public:
    User(const QString &name, int age, QObject *parent = 0);
...
}
```


- The notify signal is needed for correct property bindings!
- `Q_PROPERTY` must be at top of class

Exporting C++ Objects to QML

- Class **`QQmlContext`** exports the instance to QML.

```
int main(int argc, char *argv[])
{
    QGuiApplication app(argc, argv);

    BusinessLogic myBusiness;
    QQmlApplicationEngine engine;

    engine.rootContext()->setContextProperty("myBusinessLogic", &myBusiness)
    engine.load("qrc:/main.qml");

    return app.exec();
}
```

Exporting C++ Objects to QML through Registration

```
#include "businessLogic.h"
#include <QGuiApplication>
#include <qqml.h> // for qmlRegisterType
#include <QQmlApplicationEngine>

int main(int argc, char *argv[]) {
    QGuiApplication app(argc, argv);

    // Expose the class
    qmlRegisterType<BusinessLogic>("CustomComponents", 1, 0, "BL");

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    return app.exec();
}
```

Using the Class

- In the **`main.qml`** file:

```
import CustomComponents 1.0

Window {
    visible: true; width: 500; height: 360

   Rectangle {
          anchors.fill: parent

        BL {
                id: myBusinessLogic }

                }

    ...
}
```


qml-cpp-integration/ex-simple-timer

What Is Exported?

- Properties
- Signals
- Slots
- Methods marked with `Q_INVOKABLE`
- Enums registered with `Q_ENUM`

```
class IntervalSettings : public QObject  {
    Q_OBJECT
    Q_PROPERTY(int duration READ duration WRITE setDuration
                        NOTIFY durationChanged)

    Q_PROPERTY(Unit unit READ unit WRITE setUnit NOTIFY unitChanged)
    Q_ENUM(Unit)

public:
    enum Unit { Minutes, Seconds, MilliSeconds
    Q_INVOKABLE void login();
};
```

In the Next Sessions

- QWidgets
- Model/View