QML and Qt Quick

QML LANGUAGE

- Qt Modeling Language – user interface markup language.

  https://doc.qt.io/qt-5/qmlapplications.html

QML LANGUAGE

- Qt Modeling Language — user interface markup language.

  https://doc.qt.io/qt-5/qmlapplications.html

- Declarative language (similar to CSS and JSON): how components interact and relate with one another, dynamic property bindings.

```
Item {
    Rectangle {
        width: myRect.width
        height: 200
    }
    Rectangle {
        id: myRect
        width: 120
        height: 100 * 2
    }
}
```

QML LANGUAGE

- Qt Modeling Language — user interface markup language.

  https://doc.qt.io/qt-5/qmlapplications.html

- Declarative language (similar to CSS and JSON): how components interact and relate with one another, dynamic property bindings.

- Builtin JavaScript (inline or via included separate .js files)

```
Item {
    Rectangle {
        width: myRect.width
        height: 200
    }
    Rectangle {
        id: myRect
        width: 120
        height: 100 * 2
    }
}
```

QML LANGUAGE

- Qt Modeling Language — user interface markup language.

  https://doc.qt.io/qt-5/qmlapplications.html

- Declarative language (similar to CSS and JSON): how components interact and relate with one another, dynamic property bindings.

- Builtin JavaScript (inline or via included separate .js files)

- Can be integrated and extended

  by C++ or Python

```
Item {
    Rectangle {
        width: myRect.width
        height: 200
    }
    Rectangle {
        id: myRect
        width: 120
        height: 100 * 2
    }
}
```

EXECUTE QML

- QML/JavaScript Projects

  qml source code is loaded on demand at run-time. Just-in-time (JIT) compilation technique is used to generate machine code on the fly.

EXECUTE QML

- QML/JavaScript Projects

  qml source code is loaded on demand at run-time. Just-in-time (JIT) compilation technique is used to generate machine code on the fly.

- C++ Projects

  QML and JavaScript code can be compiled into native C++ binaries with the Qt Quick Compiler.

QML BENEFITS FOR GUI DEVELOPMENT

- Declarative languages are more suited for defining UIs.

QML BENEFITS FOR GUI DEVELOPMENT

- Declarative languages are more suited for defining UIs.
- QML code is simpler to write, as it is less verbose than C++, and is not strongly typed (also excellent for fast GUI prototyping).

QML BENEFITS FOR GUI DEVELOPMENT

- Declarative languages are more suited for defining UIs.
- QML code is simpler to write, as it is less verbose than C++, and is not strongly typed (also excellent for fast GUI prototyping).
- Rapid development cycles without the traditional C++ compilation steps.

QML BENEFITS FOR GUI DEVELOPMENT

- Declarative languages are more suited for defining UIs.
- QML code is simpler to write, as it is less verbose than C++, and is not strongly typed (also excellent for fast GUI prototyping).
- Rapid development cycles without the traditional C++ compilation steps.
- JavaScript can easily be used in QML to respond to events.

QT UI MODULES: 2 WAYS FOR WRITING GUI

- Qt Widgets (for C++, Qt for Python)

  https://doc.qt.io/qt-5/qtwidgets-index.html

  QCheckBox, QLabel, QSlider, QTabBar, QTableView…

QT UI MODULES: 2 WAYS FOR WRITING GUI

- Qt Widgets (for C++, Qt for Python)

  https://doc.qt.io/qt-5/qtwidgets-index.html

  QCheckBox, QLabel, QSlider, QTabBar, QTableView…

- Qt Quick (for QML)

  https://doc.qt.io/qt-5/qtquick-index.html

  CheckBox, Label, Slider, TabBar, TableView…

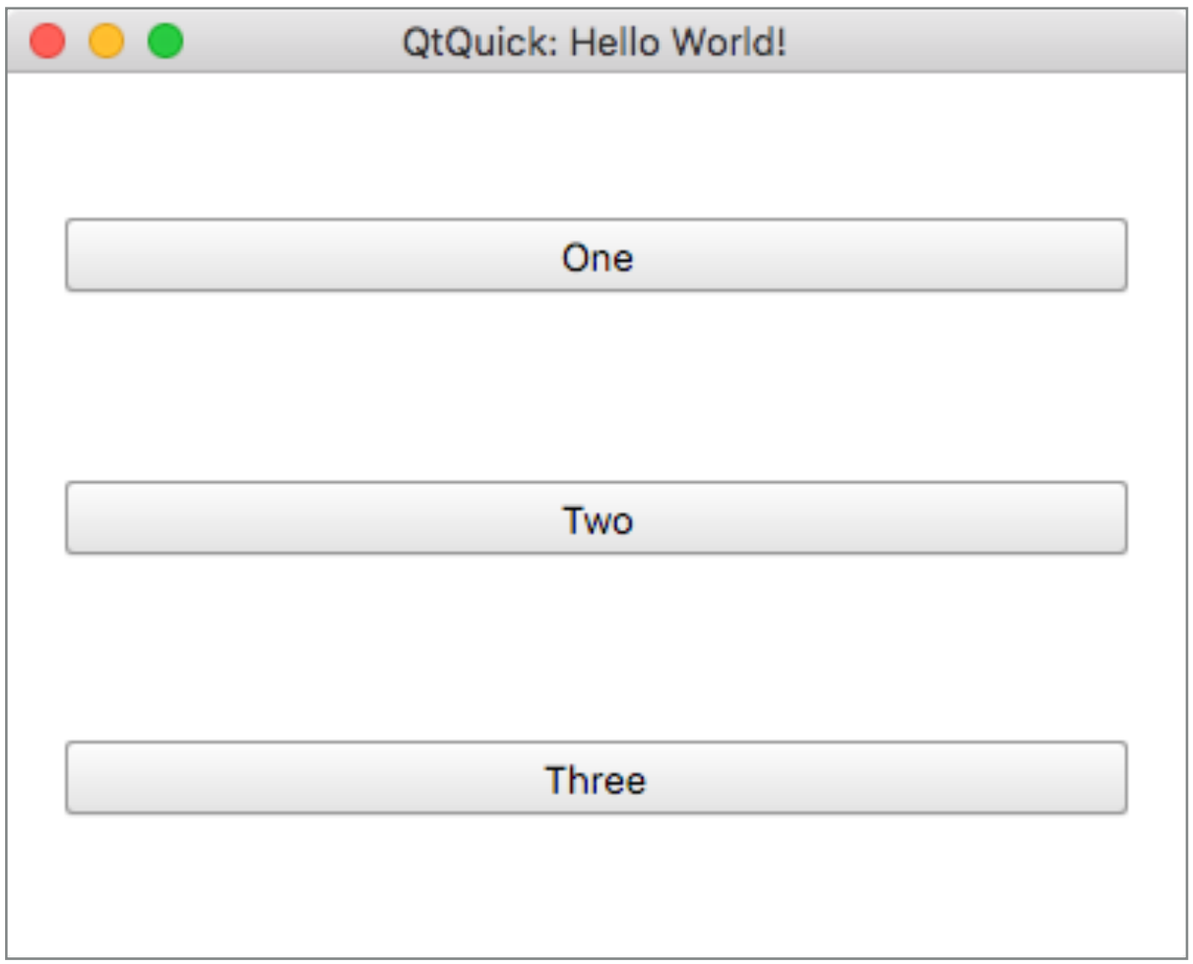
Qt Widgets vs Qt Quick

(C++ project)

Qt Widgets

Qt Widgets

Qt Quick



Qt Widgets

main.cpp

```
#include "window.h"
#include <QApplication>

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

    Window window;
    window.show();

    return app.exec();
}
```

Qt Quick

main.cpp

```
#include <QApplication>
#include <QQmlApplicationEngine>

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

    QQmlApplicationEngine engine;
    engine.load(“qrc:/window.qml");

   return app.exec();
}
```

Qt Widgets

window.h

```
#ifndef WINDOW_H
#define WINDOW_H

#include <QWidget>

class Window : public QWidget
{
    Q_OBJECT

public:
    Window(QWidget *parent = nullptr);
};

#endif // WINDOW_H
```

Qt Quick

–

Qt Widgets

window.cpp

```
#include <QVBoxLayout>
#include <QPushButton>
#include "window.h"

Window::Window(QWidget *parent)
    : QWidget(parent)
{
    resize(400, 300);
    setWindowTitle("QtWidgets: Hello World!");

    QVBoxLayout *layout = new QVBoxLayout;

    QPushButton *one = new QPushButton("One");
    layout->addWidget(one);

    QPushButton *two = new QPushButton("Two");
    layout->addWidget(two);

    QPushButton *three = new QPushButton("Three");
    layout->addWidget(three);

    setLayout(layout);
}
```

Qt Quick

window.qml

```
import QtQuick 2.0
import QtQuick.Controls 1.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0

Window {
    visible: true
    width: 400
    height: 300
    title: "QtQuick: Hello World!"

    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 20

        Button { text: "One" }
        Button { text: "Two" }
        Button { text: “Three" }
    }
}
```

Qt Widgets vs Qt Quick in Qt Designer (C++ project)

Qt Widgets

main.cpp

```
#include "window.h"
#include "ui_window.h"

Window::Window(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Window)
{
    ui->setupUi(this);
}

Window::~Window()
{
    delete ui;
}
```

Qt Quick

main.cpp

```
#include <QApplication>
#include <QQmlApplicationEngine>

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

    QQmlApplicationEngine engine;
    engine.load("qrc:/window.qml");

   return app.exec();
}
```

Qt Widgets

window.h

```
#ifndef WINDOW_H
#define WINDOW_H

#include <QWidget>

namespace Ui {
class Window;
}

class Window : public QWidget
{
    Q_OBJECT

public:
    explicit Window(QWidget *parent =
                    nullptr);
    ~Window();

private:
    Ui::Window *ui;
};

#endif // WINDOW_H
```

Qt Quick

–

Qt Widgets

window.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Window</class>
 <widget class="QWidget" name="Window">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>QtWidgets: Hello World!</string>
  </property>
  <widget class="QWidget" name="verticalLayoutWidget">
   <property name="geometry">
    <rect>
     <x>-1</x>
     <y>-1</y>
     <width>401</width>
     <height>301</height>
    </rect>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout">
    <item>
     <widget class="QPushButton" name="pushButton">
      <property name="text">
       <string>One</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_2">
      <property name="text">
       <string>Two</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_3">
      <property name="text">
       <string>Three</string>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

Qt Quick

windows.qml

```
import QtQuick 2.0
import QtQuick.Controls 1.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0

Window {
    visible: true
    width: 400
    height: 300
    title: "QtQuick: Hello World!"

    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 20

        Button { text: "One" }
        Button { text: "Two" }
        Button { text: “Three" }
    }
}
```

Qt Widgets

window.ui

```
<?xml ve
<ui vers
 <class>
 <widget
  <prope
   <rect
    <x>0
    <y>0
    <wid
    <hei
   </rec
  </prop
  <prope
   <stri
  </prop
  <widge
   <prop
    <rect>
     <x>-1</x>
     <y>-1</y>
     <width>401</width>
     <height>301</height>
    </rect>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout">
    <item>
     <widget class="QPushButton" name="pushButton">
      <property name="text">
       <string>One</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_2">
      <property name="text">
       <string>Two</string>
      </property>
     </widget>
    </item>
    <item>
     <widget class="QPushButton" name="pushButton_3">
      <property name="text">
       <string>Three</string>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </widget>
 <layoutdefault spacing="6" margin="11"/>
 <resources/>
 <connections/>
</ui>
```

+ window.cpp

```
#include "window.h"
#include "ui_window.h"

Window::Window(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Window)
{
    ui->setupUi(this);
}
```

Qt Quick

windows.qml

```
import QtQuick 2.0
import QtQuick.Controls 1.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0

Window {
    visible: true
    width: 400
    height: 300
    title: "QtQuick: Hello World!"

    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 20

        Button { text: "One" }
        Button { text: "Two" }
        Button { text: “Three" }
    }
}
```

C++ vs Python Project (QML)

C++ project

main.cpp

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>

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

    QQmlApplicationEngine engine;
    engine.load("qrc:/window.qml");

    if (engine.rootObjects().isEmpty())
        return -1;

    return app.exec();
}
```

C++ project

main.cpp

```cpp
#include <QGuiApplication>
#include <QQmlApplicationEngine>

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

    QQmlApplicationEngine engine;
    engine.load("qrc:/window.qml");

    if (engine.rootObjects().isEmpty())
        return -1;

    return app.exec();
}
```

Python project

main.py

```python
import sys
from PySide2.QtWidgets import QApplication
from PySide2.QtQml import QQmlApplicationEngine

if __name__ == "__main__":
    app = QApplication(sys.argv)

    engine = QQmlApplicationEngine()
    engine.load("window.qml")

    if not engine.rootObjects():
        sys.exit(-1)

    sys.exit(app.exec_())
```

Load and Display QML
(without C++/Python project)

qmlscene

Qt 5 includes qmlscene, a utility that loads and displays QML documents
https://doc.qt.io/qt-5/qtquick-qmlscene.html

qmlscene

Qt 5 includes qmlscene, a utility that loads and displays QML documents
https://doc.qt.io/qt-5/qtquick-qmlscene.html

main.qml

```
import QtQuick 2.12

Rectangle {
    width: 200
    height: 100
    color: "lightblue"

    Text {
        anchors.centerIn: parent
        text: "Hello, World!"
    }
}
```

qmlscene

Qt 5 includes qmlscene, a utility that loads and displays QML documents
https://doc.qt.io/qt-5/qtquick-qmlscene.html

main.qml

```
import QtQuick 2.12

Rectangle {
    width: 200
    height: 100
    color: "lightblue"

    Text {
        anchors.centerIn: parent
        text: "Hello, World!"
    }
}
```

terminal

```
$~/Qt/5.13.0/clang_64/bin/qmlscene main.qml
```

qmlscene

Qt 5 includes qmlscene, a utility that loads and displays QML documents
https://doc.qt.io/qt-5/qtquick-qmlscene.html

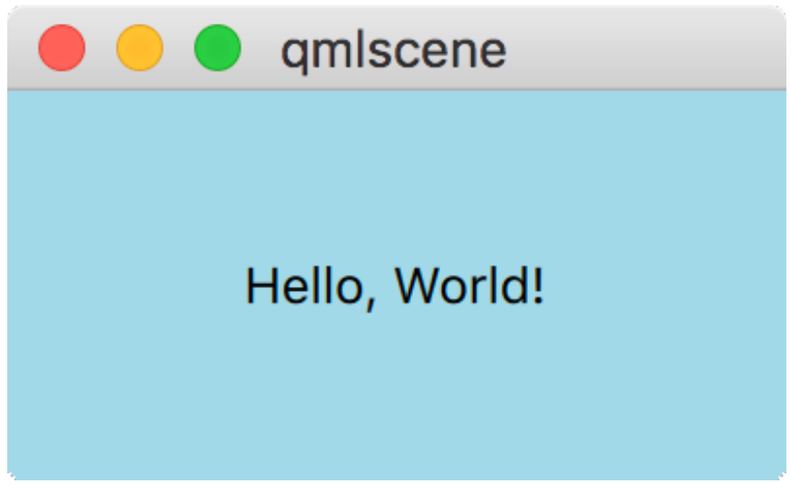
main.qml

```
import QtQuick 2.12

Rectangle {
    width: 200
    height: 100
    color: "lightblue"

    Text {
        anchors.centerIn: parent
        text: "Hello, World!"
    }
}
```

terminal

```
$~/Qt/5.13.0/clang_64/bin/qmlscene main.qml
```

result



qmlproject

qml-only project file for Qt Creator

qmlproject

qml-only project file for Qt Creator

project.qmlproject

```
import QmlProject 1.1

Project {
    mainFile: "Example1.qml"

    /* Include .qml and .js files from current directory and subdirectories */
    QmlFiles {
        directory: "."
    }
    JavaScriptFiles {
        directory: "."
    }
}
```

qmlproject

qml-only project file for Qt Creator

project.qmlproject

```
import QmlProject 1.1

Project {
    mainFile: "Example1.qml"

    /* Include .qml and .js files from current directory and subdirectories */
    QmlFiles {
        directory: "."
    }
    JavaScriptFiles {
        directory: "."
    }
}
```

Example.qml

```
import QtQuick 2.12

Rectangle {
    width: 200
    height: 100
    color: "lightblue"

    Text {
        anchors.centerIn: parent
        text: "Hello, World!"
    }
}
```

QML Code Examples

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example2.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12

Window {
    property int window_width: 400
    property int window_height: 200

    visible: true
    title: "Hello World"
    color: “#fafafa"

    width: window_width
    height: window_height

    Button {
        anchors.centerIn: parent
        text: "Window size: " + window_width + " x " + window_height
    }
}
```

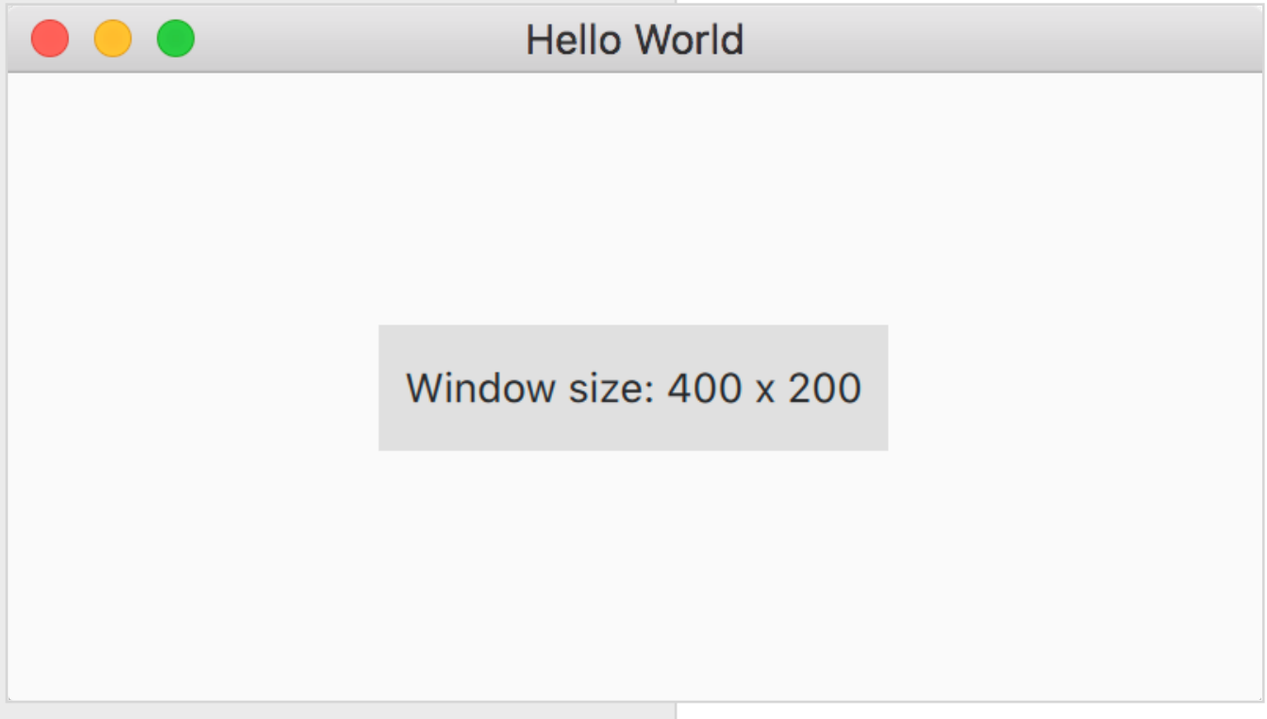
Qt Quick Controls QML Types https://doc.qt.io/qt-5/qtquick-controls2-qmlmodule.html

QML Basic Types https://doc.qt.io/qt-5/qtqml-typesystem-basictypes.html

Property Attributes https://doc.qt.io/qt-5/qtqml-syntax-objectattributes.html#property-attributes

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example2.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12

Window {
    property int window_width: 400
    property int window_height: 200

    visible: true
    title: "Hello World"
    color: “#fafafa"

    width: window_width
    height: window_height

    Button {
        anchors.centerIn: parent
        text: "Window size: " + window_width + " x " + window_height
    }
}
```



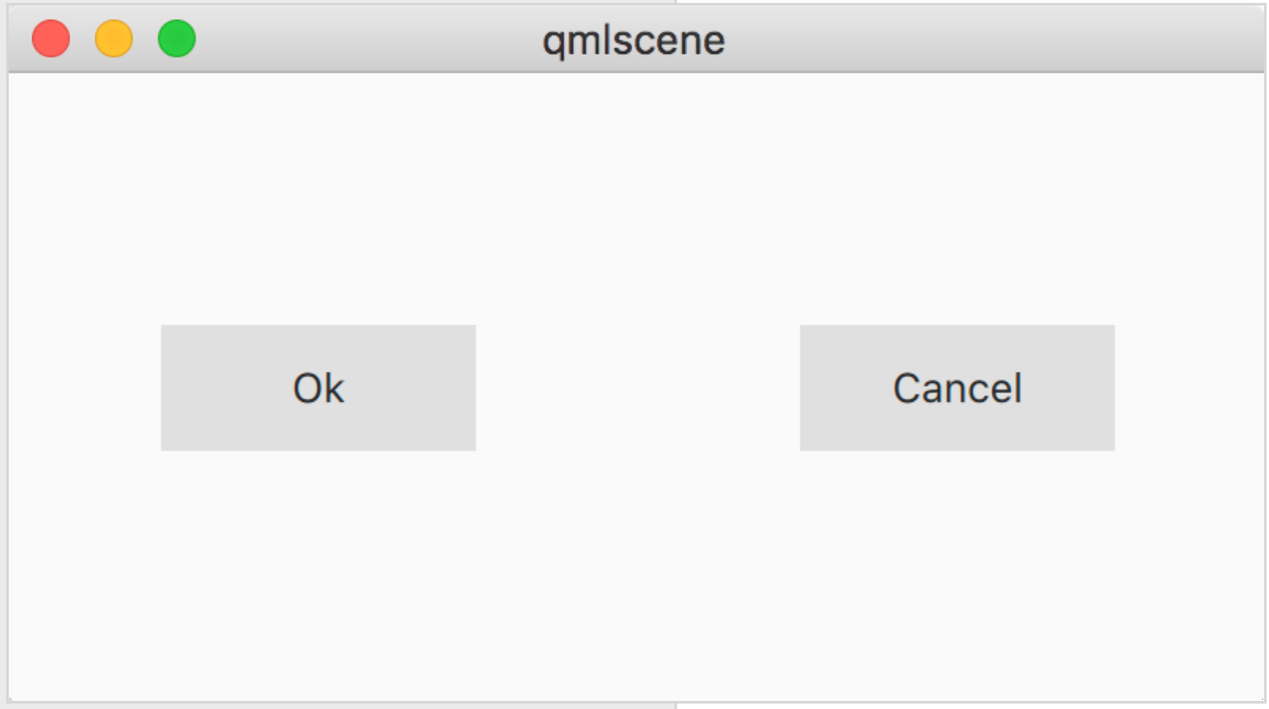
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example3.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 400
    height: 200
    color: "#fafafa"

    RowLayout {
        anchors.fill: parent

        Button {
            Layout.alignment: Qt.AlignCenter
            text: "Ok"
        }

        Button {
            Layout.alignment: Qt.AlignCenter
            text: "Cancel"
            ToolTip.visible: hovered
            ToolTip.text: qsTr("This is a tooltip message")
        }
    }
}
```

Qt Quick Layouts QML Types https://doc.qt.io/qt-5/qtquick-layouts-qmlmodule.html

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example3.qml

```qml
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 400
    height: 200
    color: "#fafafa"

    RowLayout {
        anchors.fill: parent

        Button {
            Layout.alignment: Qt.AlignCenter
            text: "Ok"
        }

        Button {
            Layout.alignment: Qt.AlignCenter
            text: "Cancel"
            ToolTip.visible: hovered
            ToolTip.text: qsTr("This is a tooltip message")
        }
    }
}
```



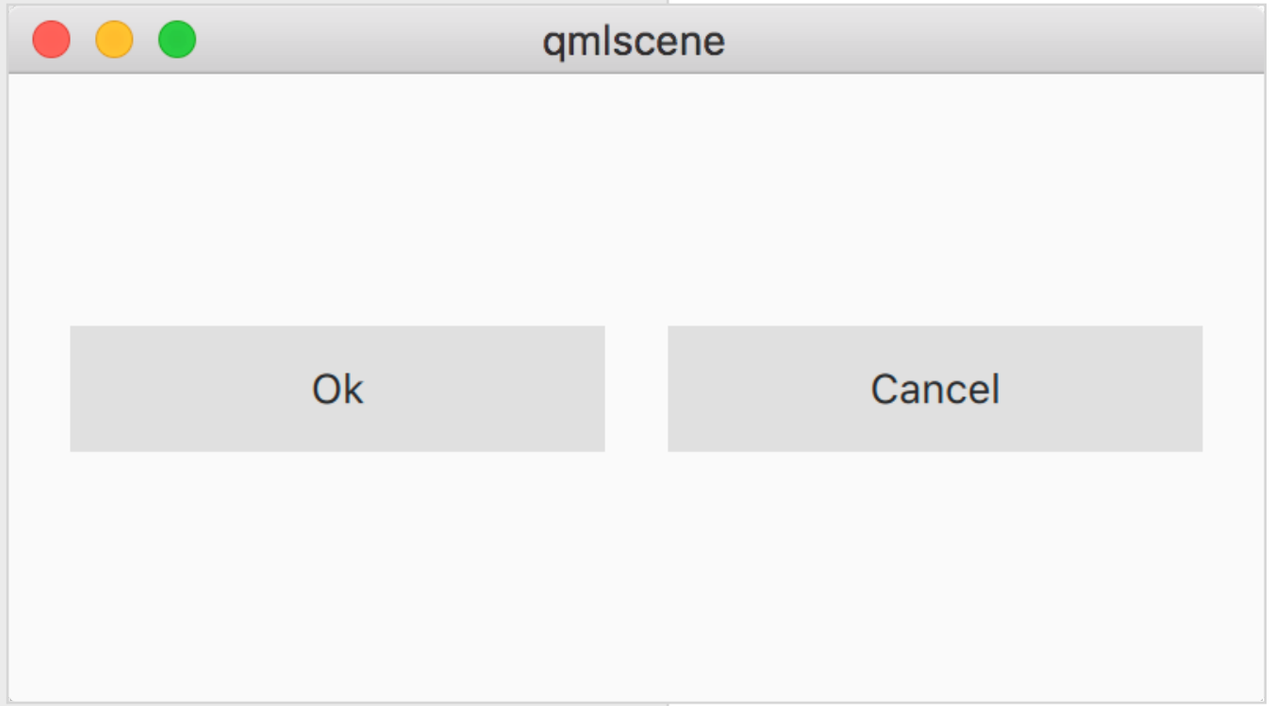
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example4.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 400
    height: 200
    color: "#fafafa"

    RowLayout {
        anchors.fill: parent
        anchors.margins: 20
        spacing: 20

        Button {
            Layout.fillWidth: true
            id: okButton
            text: "Ok"
        }

        Button {
            Layout.fillWidth: true
            text: "Cancel"
            onClicked: {
                okButton.text = "Not Ok!!!"
            }
        }
    }
}
```

Signal and Handler Event System https://doc.qt.io/qt-5/qtqml-syntax-signals.html

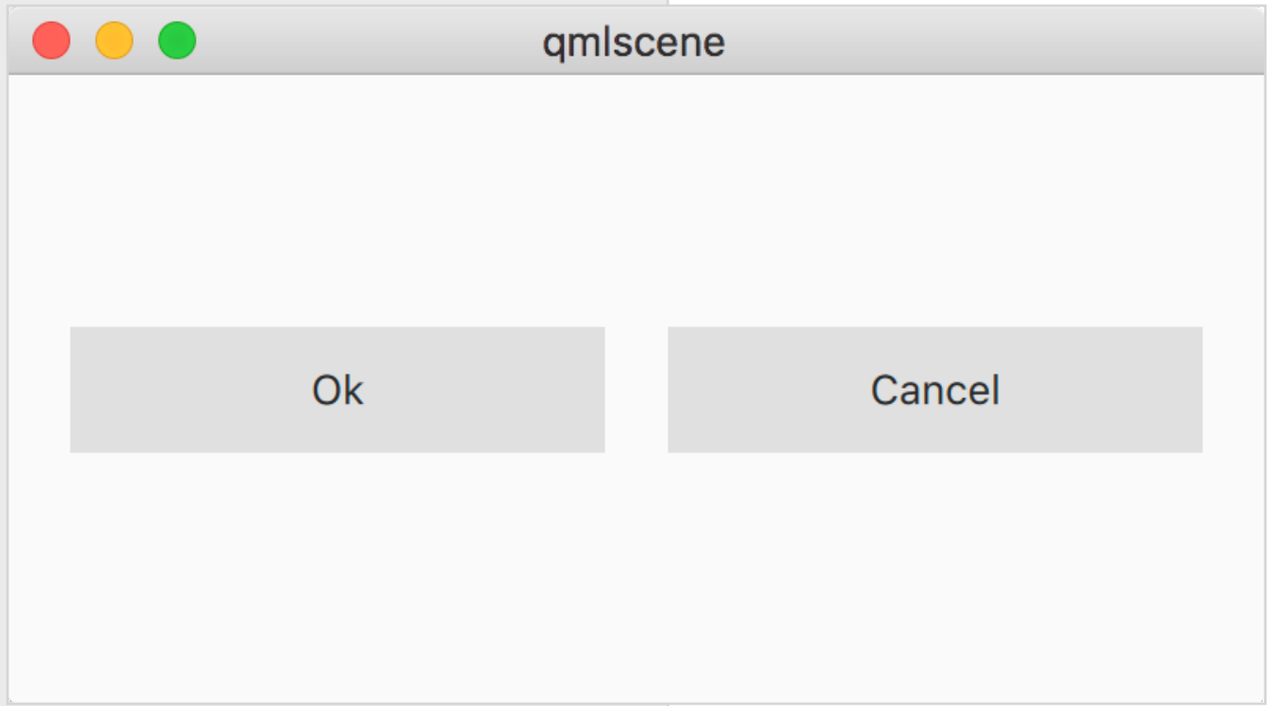
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example4.qml

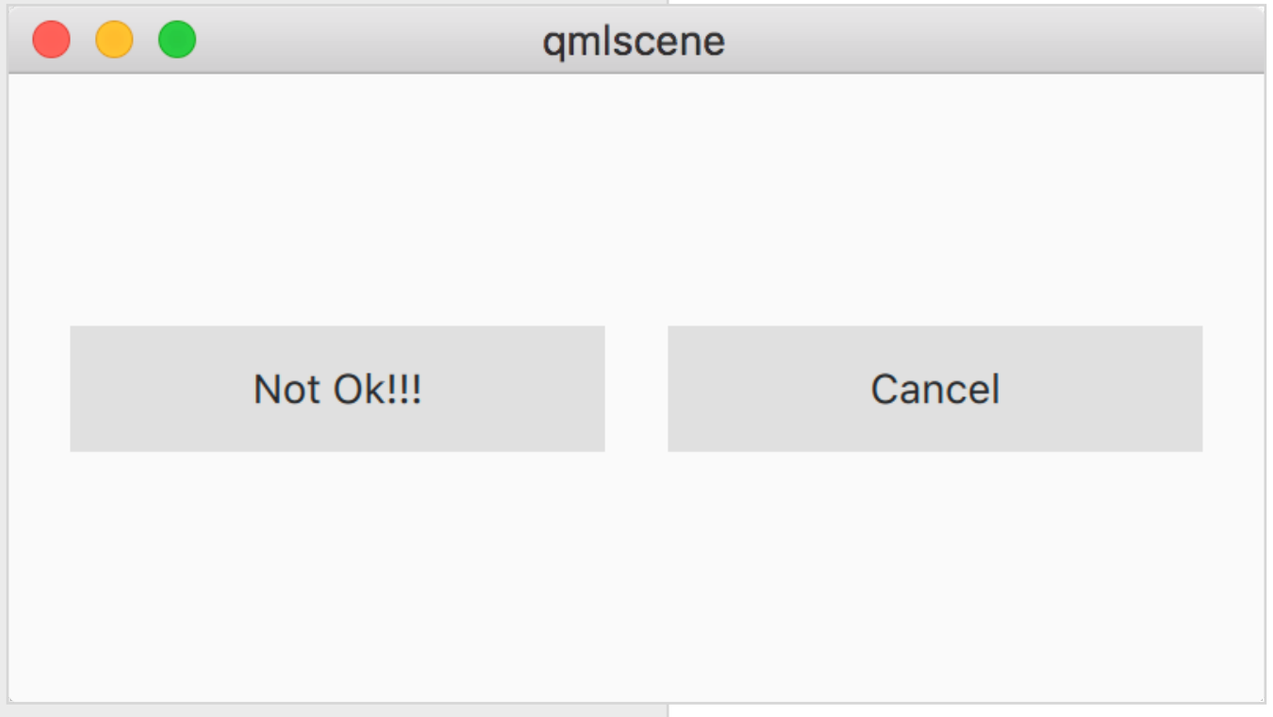
```qml
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 400
    height: 200
    color: "#fafafa"

    RowLayout {
        anchors.fill: parent
        anchors.margins: 20
        spacing: 20

        Button {
            Layout.fillWidth: true
            id: okButton
            text: "Ok"
        }

        Button {
            Layout.fillWidth: true
            text: "Cancel"
            onClicked: {
                okButton.text = "Not Ok!!!"
            }
        }
    }
}
```



https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example4.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 400
    height: 200
    color: "#fafafa"

    RowLayout {
        anchors.fill: parent
        anchors.margins: 20
        spacing: 20

        Button {
            Layout.fillWidth: true
            id: okButton
            text: "Ok"
        }

        Button {
            Layout.fillWidth: true
            text: "Cancel"
            onClicked: {
                okButton.text = "Not Ok!!!"
            }
        }
    }
}
```





https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example5.qml

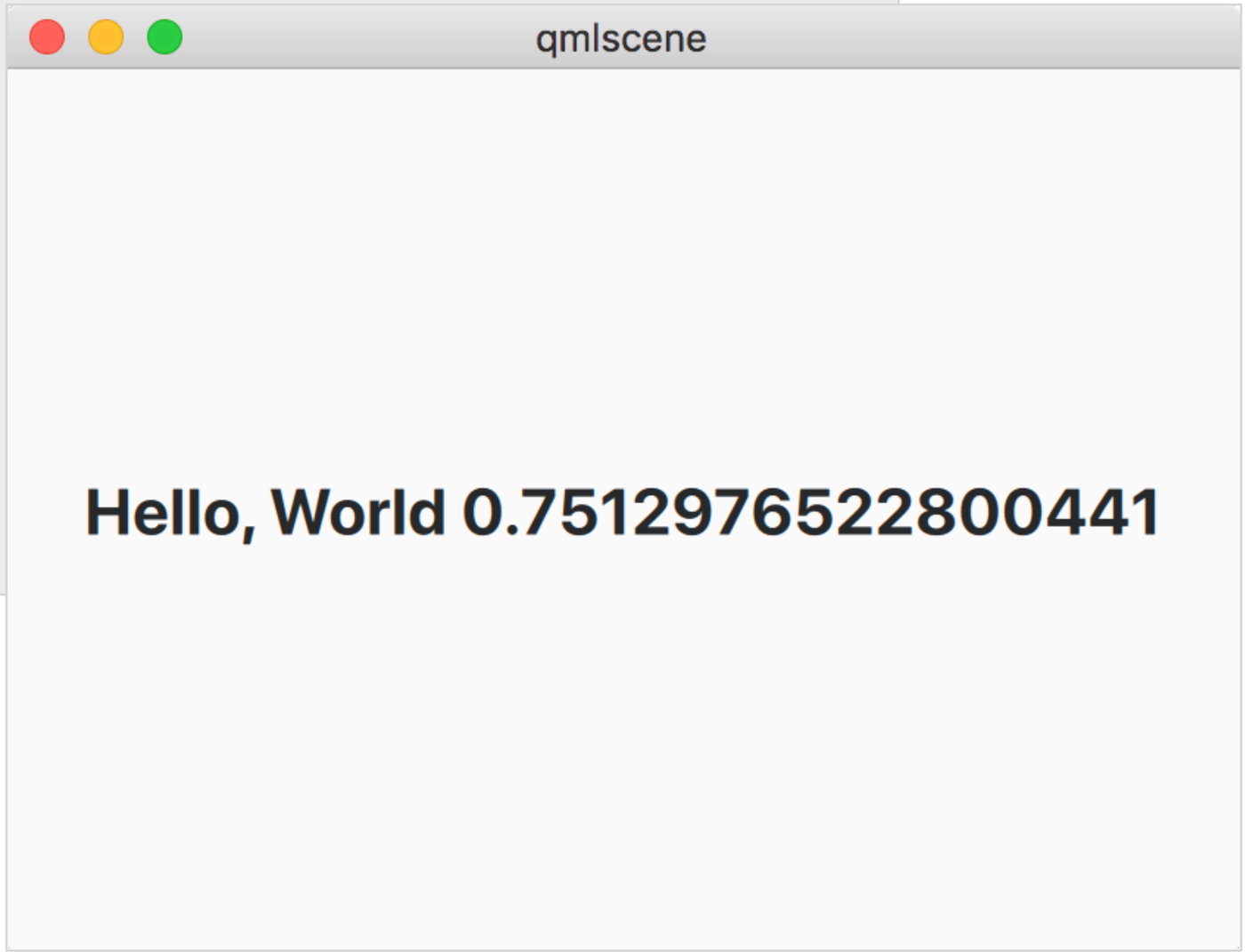
```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12

Window {
    visible: true
    width: 420
    height: 300
    color: "#fafafa"

    Label {
        id: label
        anchors.centerIn: parent
        font.pixelSize: 22
        font.bold: true
        text: "Hello, World "
    }

    Component.onCompleted: {
        label.text += Math.random()
    }
}
```

Signal and Handler Event System https://doc.qt.io/qt-5/qml-qtqml-component.html#completed-signal

JavaScript in application startup code https://doc.qt.io/qt-5/qtqml-javascript-expressions.html#javascript-in-application-startup-code

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example5.qml

```qml
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12

Window {
    visible: true
    width: 420
    height: 300
    color: "#fafafa"

    Label {
        id: label
        anchors.centerIn: parent
        font.pixelSize: 22
        font.bold: true
        text: "Hello, World "
    }

    Component.onCompleted: {
        label.text += Math.random()
    }
}
```



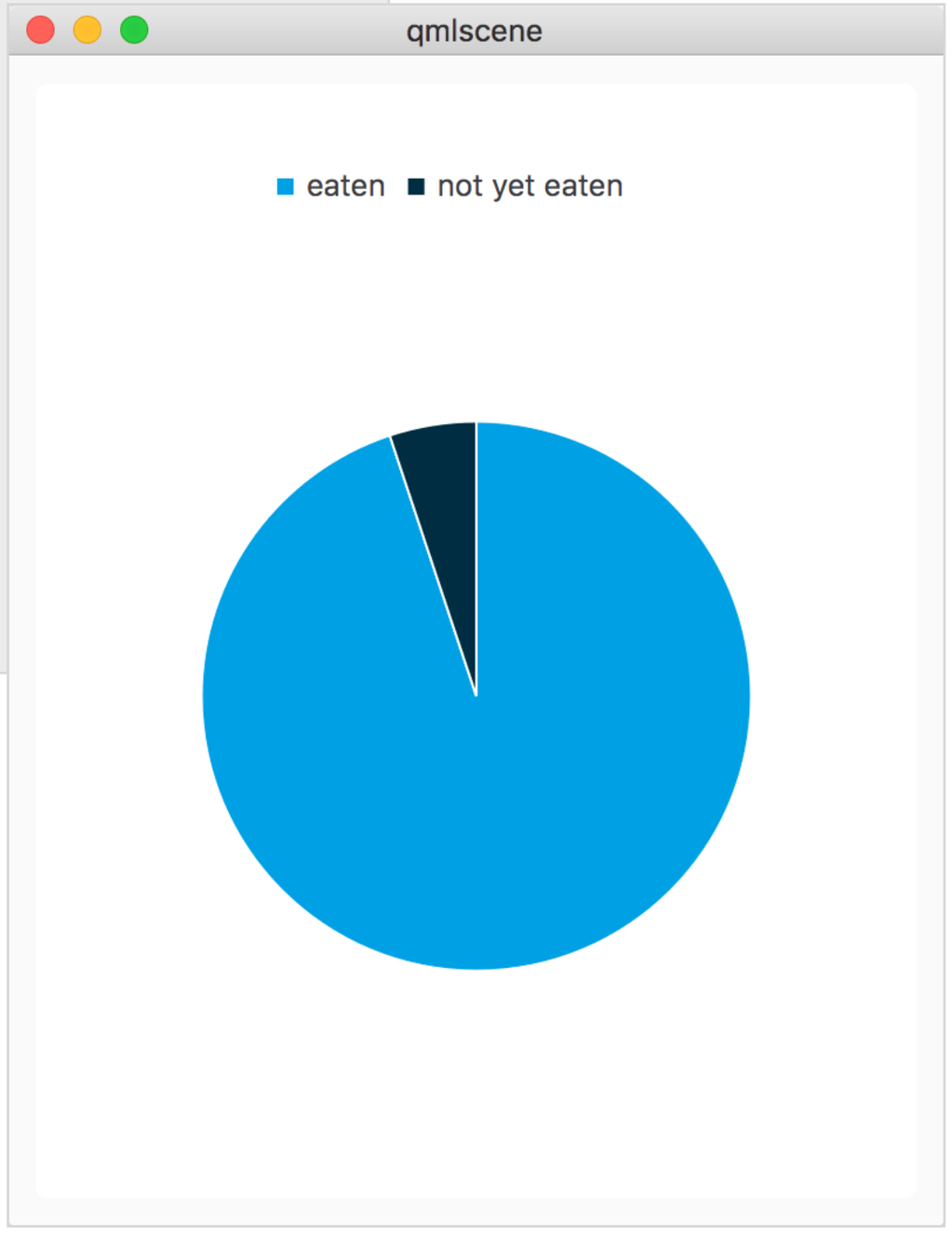
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example6.qml

```qml
import QtQuick 2.12
import QtQuick.Window 2.12
import QtCharts 2.3

Window {
    visible: true
    width: 400
    height: 500
    color: "#fafafa"

    ChartView {
        anchors.fill: parent
        antialiasing: true

        PieSeries {
            PieSlice { label: "eaten"; value: 94.9 }
            PieSlice { label: "not yet eaten"; value: 5.1 }
        }
    }
}
```

Qt Charts QML Types https://doc.qt.io/qt-5/qtcharts-qmlmodule.html

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example6.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtCharts 2.3

Window {
    visible: true
    width: 400
    height: 500
    color: "#fafafa"

    ChartView {
        anchors.fill: parent
        antialiasing: true

        PieSeries {
            PieSlice { label: "eaten"; value: 94.9 }
            PieSlice { label: "not yet eaten"; value:
        }
    }
}
```



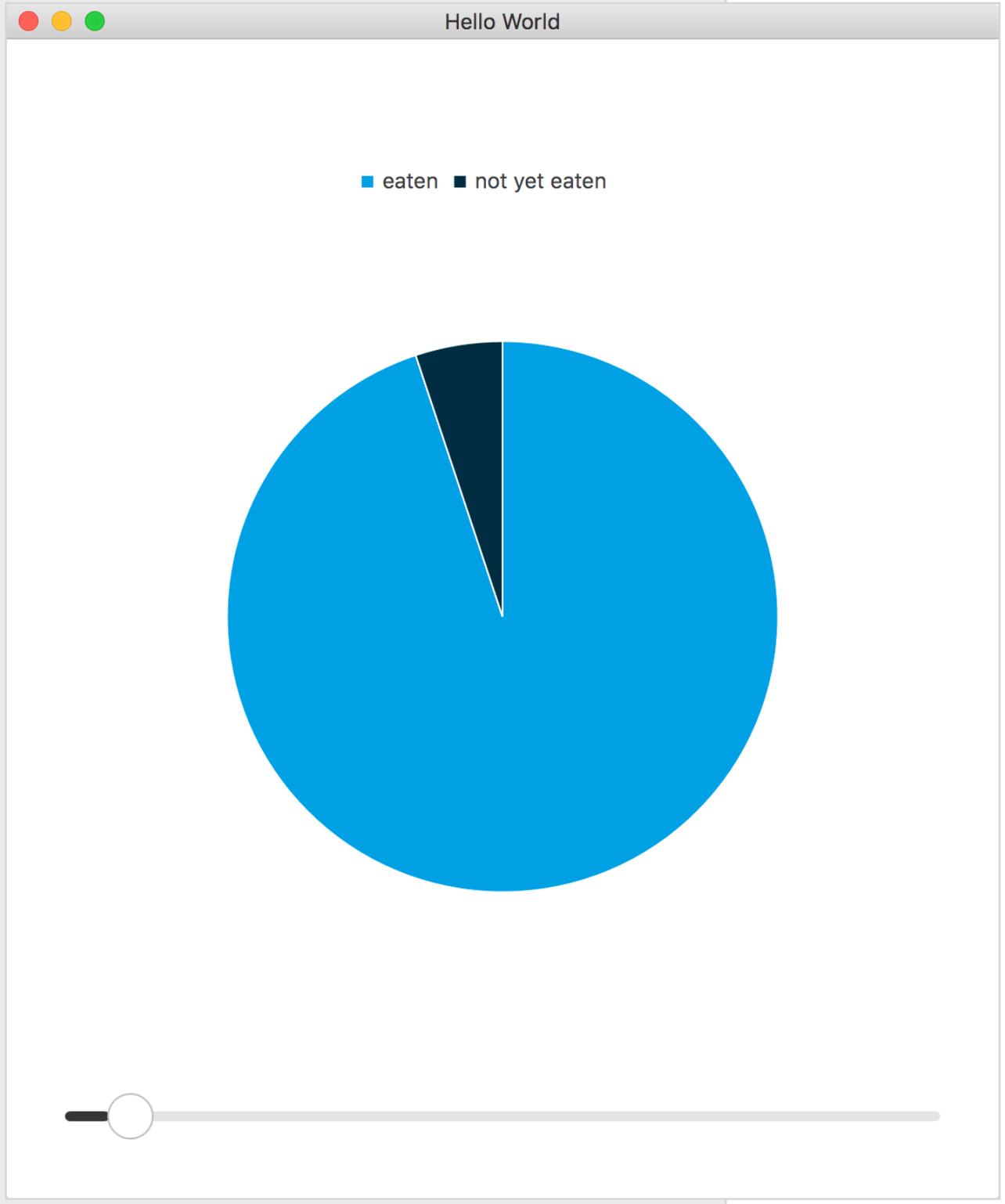
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example7.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12
import QtCharts 2.3

Window {
    visible: true
    width: 600; height: 700

    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 30
        spacing: 10

        ChartView {
            Layout.fillWidth: true
            Layout.fillHeight: true
            PieSeries {
                PieSlice { label: "eaten"; value: 100 - slider.value }
                PieSlice { label: "not yet eaten"; value: slider.value }
            }
        }

        Slider {
            id: slider
            Layout.fillWidth: true
            from: 0; value: 5.1; to: 100
        }
    }
}
```

Property Binding https://doc.qt.io/qt-5/qtqml-syntax-propertybinding.html

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example7.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12
import QtCharts 2.3

Window {
    visible: true
    width: 600; height: 700

    ColumnLayout {
        anchors.fill: parent
        anchors.margins: 30
        spacing: 10

        ChartView {
            Layout.fillWidth: true
            Layout.fillHeight: true
            PieSeries {
                PieSlice { label: "eaten";
                PieSlice { label: "not yet
            }
        }

        Slider {
            id: slider
            Layout.fillWidth: true
            from: 0; value: 5.1; to: 100
        }
    }
}
```



https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example8.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 600; height: 400

    ColumnLayout {
        anchors.fill: parent
        spacing: 0

        TabBar {
            id: tabBar
            Layout.fillWidth: true

            TabButton { text: qsTr("darkseagreen") }
            TabButton { text: qsTr("lightblue") }
        }

        StackLayout {
            Layout.fillWidth: true
            Layout.fillHeight: true
            currentIndex: tabBar.currentIndex

            Rectangle { color: "darkseagreen" }
            Rectangle { color: "lightblue" }
        }
    }
}
```

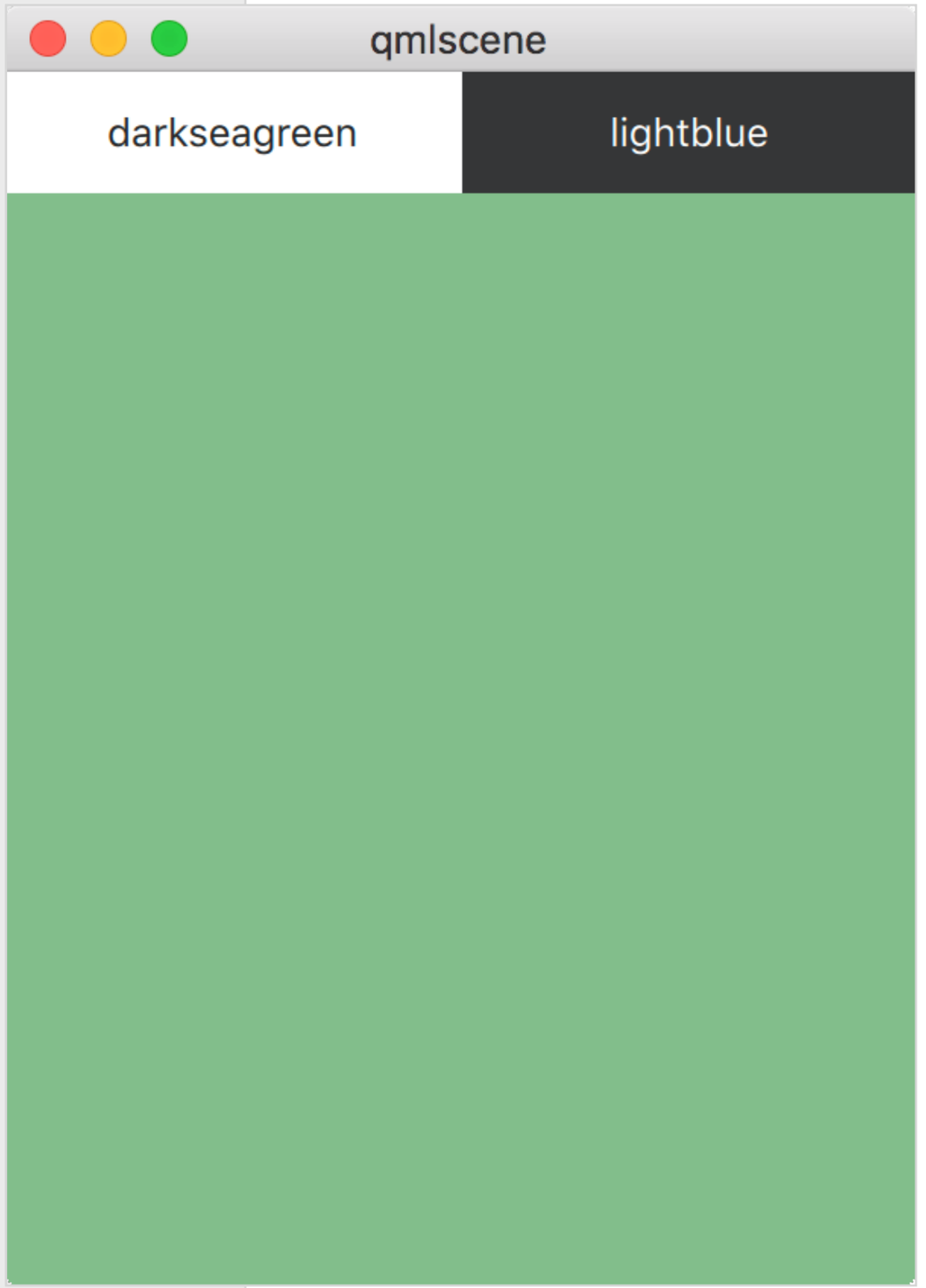
TabBar QML Type https://doc.qt.io/qt-5/qml-qtquick-controls2-tabbar.html

StackLayout QML Type https://doc.qt.io/qt-5/qml-qtquick-layouts-stacklayout.html

currentIndex binding

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example8.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12
import QtQuick.Controls 2.12
import QtQuick.Layouts 1.12

Window {
    visible: true
    width: 600; height: 400

    ColumnLayout {
        anchors.fill: parent
        spacing: 0

        TabBar {
            id: tabBar
            Layout.fillWidth: true

            TabButton { text: qsTr("darkseagreen") }
            TabButton { text: qsTr("lightblue") }
        }

        StackLayout {
            Layout.fillWidth: true
            Layout.fillHeight: true
            currentIndex: tabBar.currentIndex

            Rectangle { color: "darkseagreen" }
            Rectangle { color: "lightblue" }
        }
    }
}
```



https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example9.qml

Charts/SimpleChart.qml

```
import QtCharts 2.3

ChartView {
    antialiasing: true

    PieSeries {
        PieSlice { label: "eaten"; value: 94.9 }
        PieSlice { label: "not yet eaten"; value: 5.1 }
    }
}
```

Import standard module

Import statements https://doc.qt.io/qt-5/qtqml-syntax-imports.html

https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example9.qml

Charts/SimpleChart.qml

```
import QtCharts 2.3

ChartView {
    antialiasing: true

    PieSeries {
        PieSlice { label: "eaten"; value: 94.9 }
        PieSlice { label: "not yet eaten"; value: 5.1 }
    }
}
```

Import standard module

Example9.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12

import "Charts" as CustomCharts

Window {
    visible: true
    color: "#fafafa"
    width: 600
    height: 400

    CustomCharts.SimpleChart {
        anchors.fill: parent
    }
}
```

Import directory with QML documents

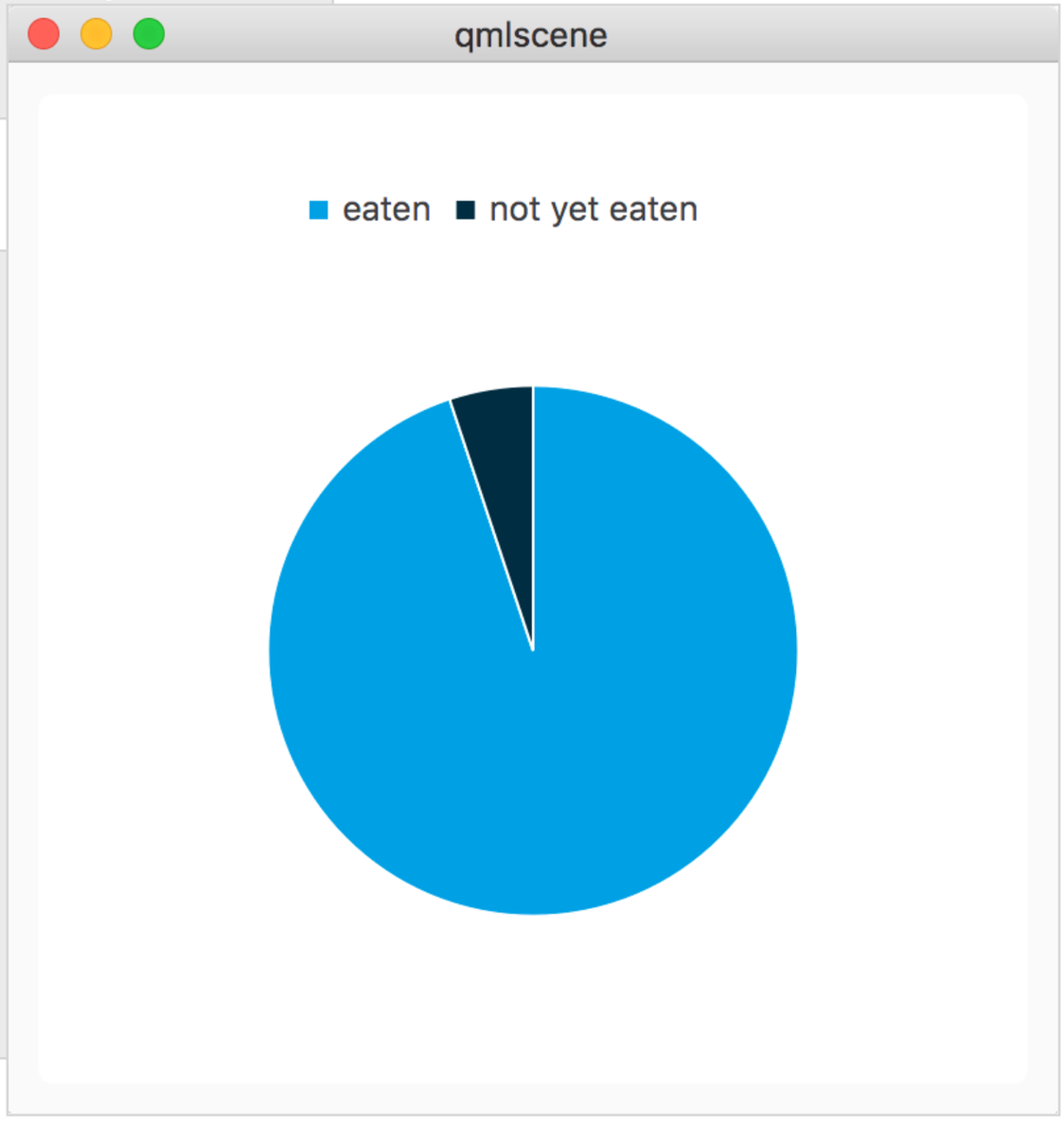
https://github.com/easyDiffraction/QmlPlayground/blob/master/Pure_Qml/Example9.qml

Charts/SimpleChart.qml

```
import QtCharts 2.3

ChartView {
    antialiasing: true

    PieSeries {
        PieSlice { label: "eaten"; value: 94.9 }
        PieSlice { label: "not yet eaten"; value: 5.1 }
    }
}
```

Example9.qml

```
import QtQuick 2.12
import QtQuick.Window 2.12

import "Charts" as CustomCharts

Window {
    visible: true
    color: "#fafafa"
    width: 600
    height: 400

    CustomCharts.SimpleChart {
        anchors.fill: parent
    }
}
```



QML Business Logic

QML BUSINESS LOGIC

- JavaScript (natively)
- C++
- Python

QML BUSINESS LOGIC

- JavaScript (natively)
- C++
- Python

JavaScript v.1 (single file)

https://github.com/easyDiffraction/QmlPlayground/blob/master/Business_Logic/Js/Qml_and_Js_in_single_file/window.qml

window.qml

```
import QtQuick 2.0
import QtQuick.Controls 2.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0

Window {
    property var hello: ["Hello World", "Hallo Welt", "Hei maailma", "Hola Mundo"]

    visible: true
    width: 200; height: 200

    ColumnLayout {
        anchors.fill: parent; anchors.margins: 20

        Label { id: label; text: "Initial label text" }

        Button {
            text: "Click me”
            onClicked: label.text = hello[Math.floor(Math.random() * hello.length)]
        }
    }
}
```

JavaScript v.2 (separate files)

https://github.com/easyDiffraction/QmlPlayground/tree/master/Business_Logic/Js/Qml_and_Js_in_separate_files

helloMessage.js

```
function randomHello() {
    const hello = ["Hello World", "Hallo Welt", "Hei maailma", "Hola Mundo"]
    return hello[Math.floor(Math.random() * hello.length)]
}
```

JavaScript v.2 (separate files)

https://github.com/easyDiffraction/QmlPlayground/tree/master/Business_Logic/Js/Qml_and_Js_in_separate_files

helloMessage.js

```
function randomHello() {
    const hello = ["Hello World", "Hallo Welt", "Hei maailma", "Hola Mundo"]
    return hello[Math.floor(Math.random() * hello.length)]
}
```

window.qml

```
import QtQuick 2.0
import QtQuick.Controls 2.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0
import "helloMessage.js" as HelloMessage

Window {
    visible: true
    width: 200; height: 200

    ColumnLayout {
        anchors.fill: parent; anchors.margins: 20

        Label { id: label; text: "Initial label text" }

        Button {
            text: "Click me"
            onClicked: label.text = HelloMessage.randomHello()
        }
    }
}
```

Import JavaScript resources

More about JavaScript Expressions in QML Documents: https://doc.qt.io/qt-5/qtqml-javascript-expressions.html

C++ logic

main.cpp

```
#include <QObject>
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQmlContext>

class HelloMessage : public QObject
{
    Q_OBJECT
public:
    Q_INVOKABLE QString randomHello() const {
    return m_hello[ qrand() % (m_hello.size() + 1) ];
}
private:
    QStringList m_hello{"Hello World", "Hallo Welt", "Hei maailma", "Hola Mundo"};
};

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

    QQmlApplicationEngine engine;
    engine.load("qrc:/window.qml");

    HelloMessage msg;
    engine.rootContext()->setContextProperty("helloMessage", &msg);

    return app.exec();
}
```

load an application from a single QML file
https://doc.qt.io/qt-5/qqmlapplicationengine.html

Register ‘msg’ object of ‘HelloMessage()’ class to be accessible from QML by the name ‘helloMessage’

Python logic

main.py

```
import os, sys, random
from PySide2.QtCore import QUrl, QObject, Signal, Slot
from PySide2.QtQml import QQmlApplicationEngine
from PySide2.QtWidgets import QApplication

class HelloMessage(QObject):
    def __init__(self):
        QObject.__init__(self)
        self.hello = ["Hello World", "Hallo Welt", "Hei maailma", "Hola Mundo"]

    @Slot(result=str)
    def randomHello(self):
        return random.choice(self.hello)

if __name__ == '__main__':
    app = QApplication(sys.argv)

    engine = QQmlApplicationEngine()
    engine.load("window.qml")

    msg = HelloMessage()
    engine.rootContext().setContextProperty("helloMessage", msg)

    sys.exit(app.exec_())
```

load an application from a single QML file
https://doc.qt.io/qt-5/qqmlapplicationengine.html

Register 'msg' object of 'HelloMessage()' class to be accessible from QML by the name 'helloMessage'

QML GUI for C++ / Python

window.qml

```
import QtQuick 2.0
import QtQuick.Controls 2.0
import QtQuick.Layouts 1.0
import QtQuick.Window 2.0

Window {
    visible: true
    width: 200; height: 200

    ColumnLayout {
        anchors.fill: parent; anchors.margins: 20

        Label { id: label; text: "Initial label text" }

        Button {
            text: "Click me"
            onClicked: label.text = helloMessage.randomHello()
        }
    }
}
```

From python: engine.rootContext().setContextProperty("helloMessage", msg)

More information about QML and QtQuick

MORE INFORMATION ABOUT QML AND QTQUICK

- Qt Quick: https://doc.qt.io/qt-5/qtquick-index.html
- Qt QML: https://doc.qt.io/qt-5/qtqml-index.html
- QML Applications: https://doc.qt.io/qt-5/qmlapplications.html
- Best Practices for QML and Qt Quick: https://doc.qt.io/qt-5/qtquick-bestpractices.html
- Performance Considerations And Suggestions: https://doc.qt.io/qt-5/qtquick-performance.html
- QtCreator > Help

MORE INFORMATION ABOUT QML AND QTQUICK

- Qt Quick: https://doc.qt.io/qt-5/qtquick-index.html
- Qt QML: https://doc.qt.io/qt-5/qtqml-index.html
- QML Applications: https://doc.qt.io/qt-5/qmlapplications.html
- Best Practices for QML and Qt Quick: https://doc.qt.io/qt-5/qtquick-bestpractices.html
- Performance Considerations And Suggestions: https://doc.qt.io/qt-5/qtquick-performance.html
- QtCreator > Help
- ... google

QML Playground

QML PLAYGROUND

- https://github.com/easyDiffraction/QmlPlayground