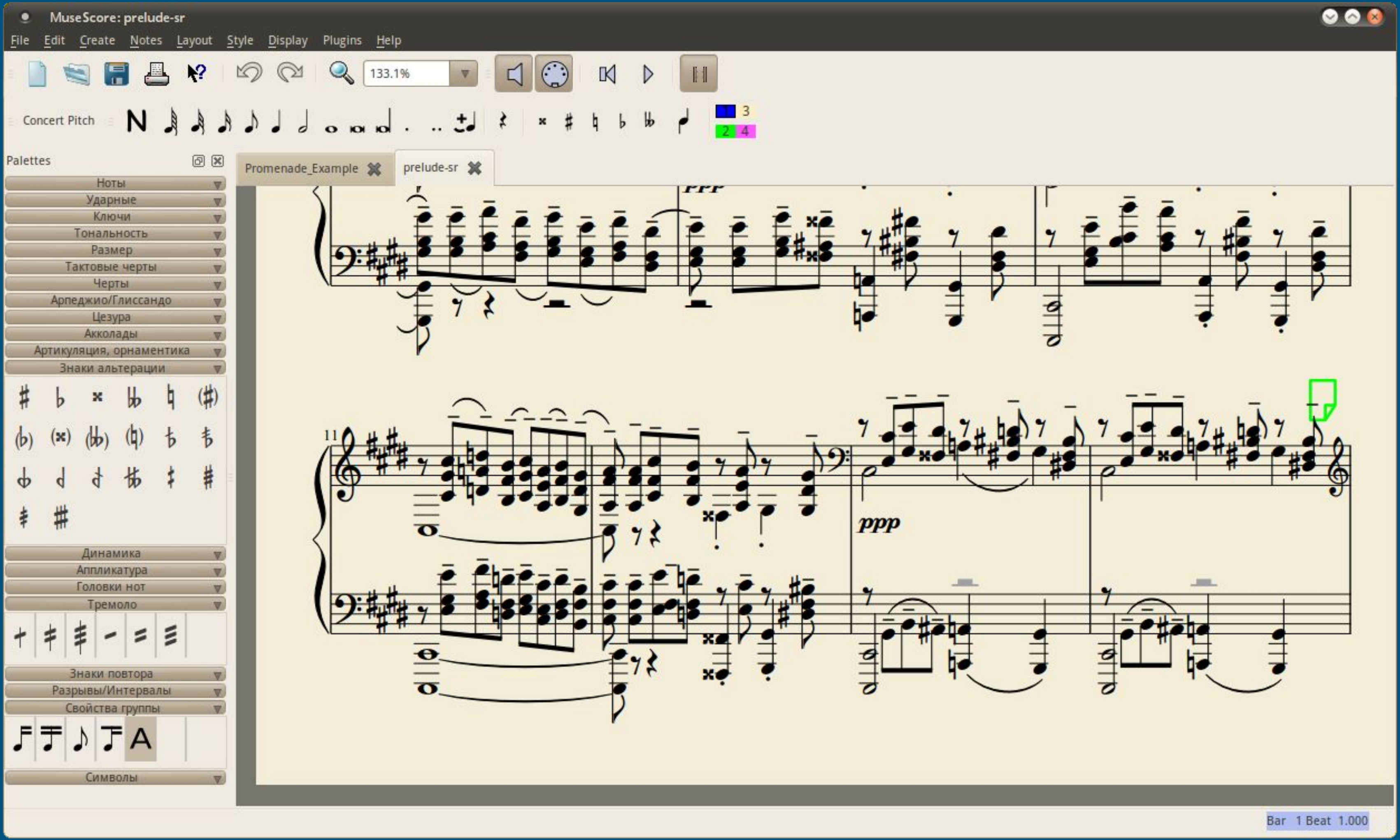
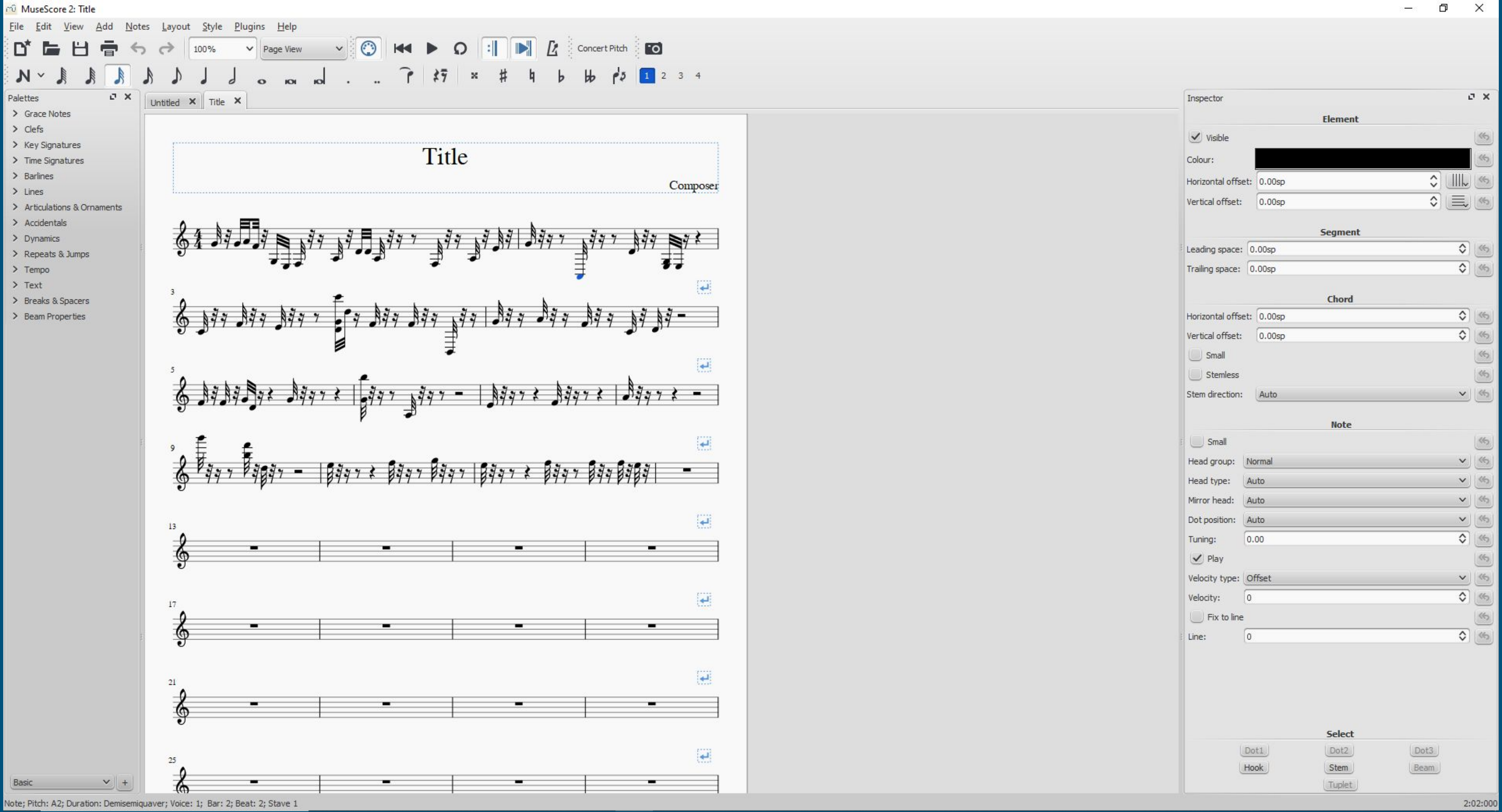
HOW TO BREATHE A NEW QML LIFE

into a QWidget-based app from 2000s


MuseScore: prelude-sr
File
Edit
Create
Notes
Layout
Style
Display
Plugins
Help
133.1%
Concert Pitch
N
1
3
2
4
Palettes
Ноты
Ударные
Ключи
Тональность
Размер
Тактовые черты
Черты
Арпеджио/Глиссандо
Цезура
Акколады
Артикуляция, орнаментика
Знаки альтерации
Динамика
Аппликатура
Головки нот
Тремоло
Знаки повтора
Разрывы/Интервалы
Свойства группы
A
Символы
Promenade_Example
prelude-sr
ppp
11
ppp
Bar 1 Beat 1.000



Inspector

Element

☑ Visible

Colour:

Horizontal offset: 0.00sp

Vertical offset: 0.00sp

Segment

Leading space: 0.00sp

Trailing space: 0.00sp

Chord

Horizontal offset: 0.00sp

Vertical offset: 0.00sp

☐ Small

☐ Stemless

Stem direction: Auto

Note

☐ Small

Head group: Normal

Head type: Auto

Mirror head: Auto

Dot position: Auto

Tuning: 0.00

☑ Play

Velocity type: Offset

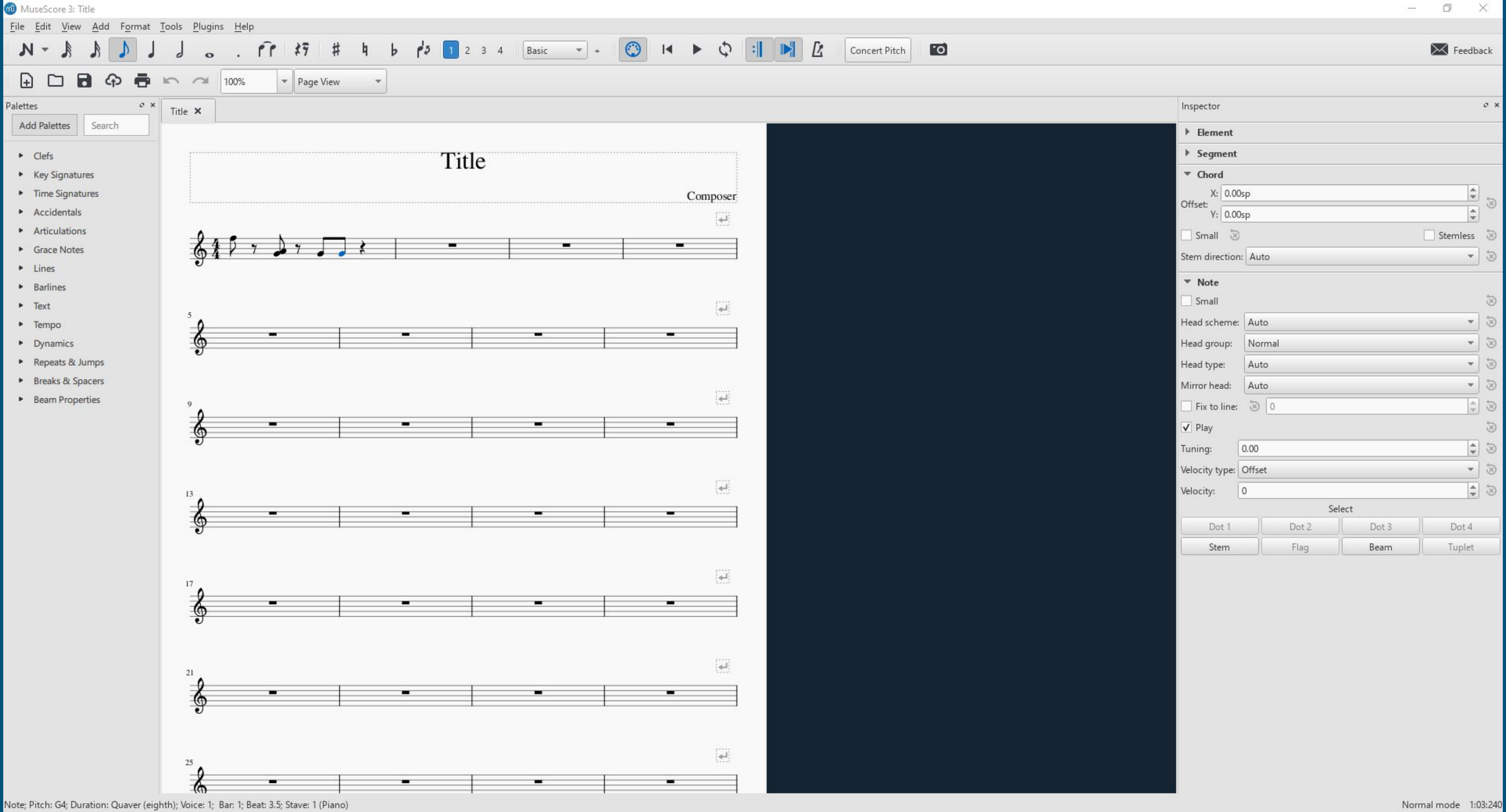
Velocity: 0

☐ Fix to line

Line: 0

Note; Pitch: A2; Duration: Demisemiquaver; Voice: 1; Bar: 2; Beat: 2; Stave 1

2:02:000

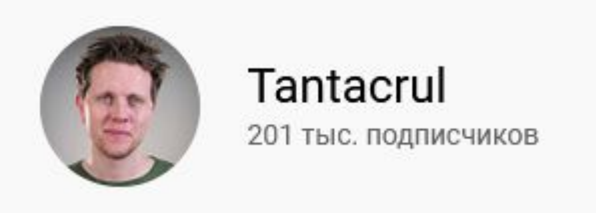


Offset: X: 0.00sp
Offset: Y: 0.00sp
□ Small
□ Stemless
Stem direction: Auto

□ Small
Head scheme: Auto
Head group: Normal
Head type: Auto
Mirror head: Auto
□ Fix to line: 0
☑ Play
Tuning: 0.00
Velocity type: Offset
Velocity: 0

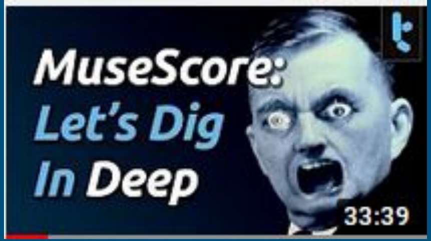
Note; Pitch: G4; Duration: Quaver (eighth); Voice: 1; Bar: 1; Beat: 3.5; Stave: 1 (Piano)

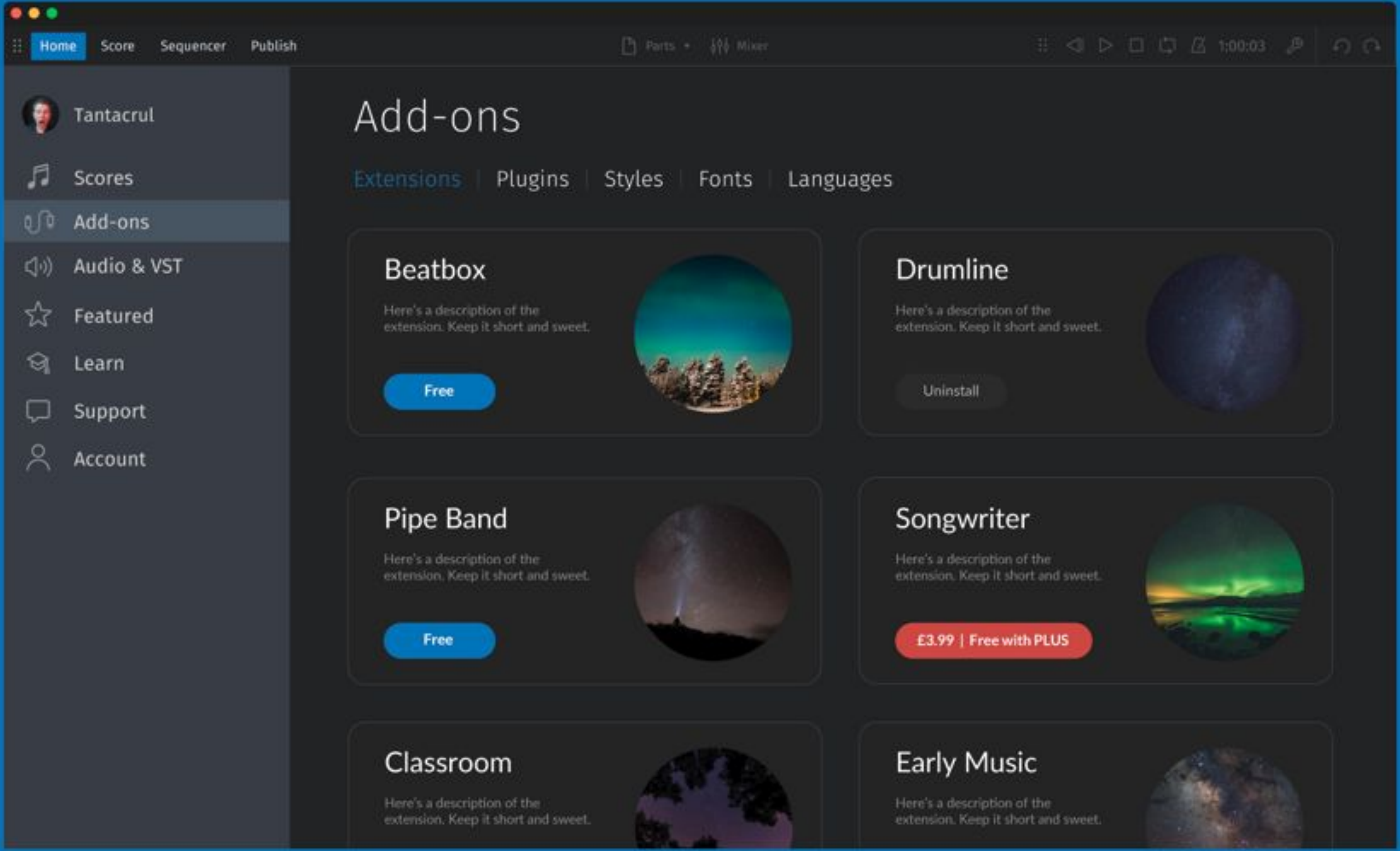
Normal mode 1:03:240



Tantacrul

201 тыс. подписчиков





Add-ons

Extensions | Plugins | Styles | Fonts | Languages

Beatbox

Here's a description of the extension. Keep it short and sweet.

Free

Drumline

Here's a description of the extension. Keep it short and sweet.

Uninstall

Pipe Band

Here's a description of the extension. Keep it short and sweet.

Free

Songwriter

Here's a description of the extension. Keep it short and sweet.

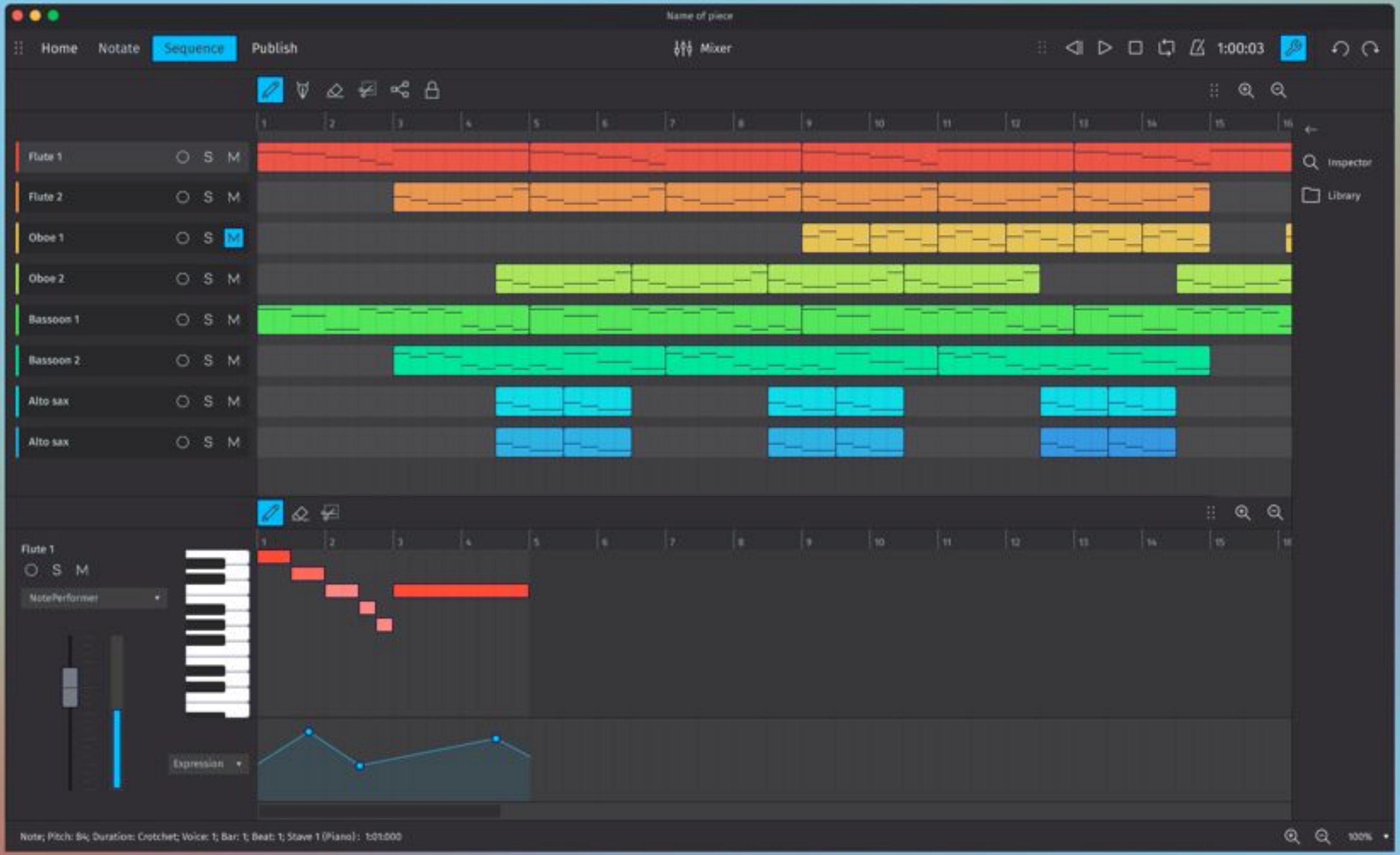
£3.99 | Free with PLUS

Classroom

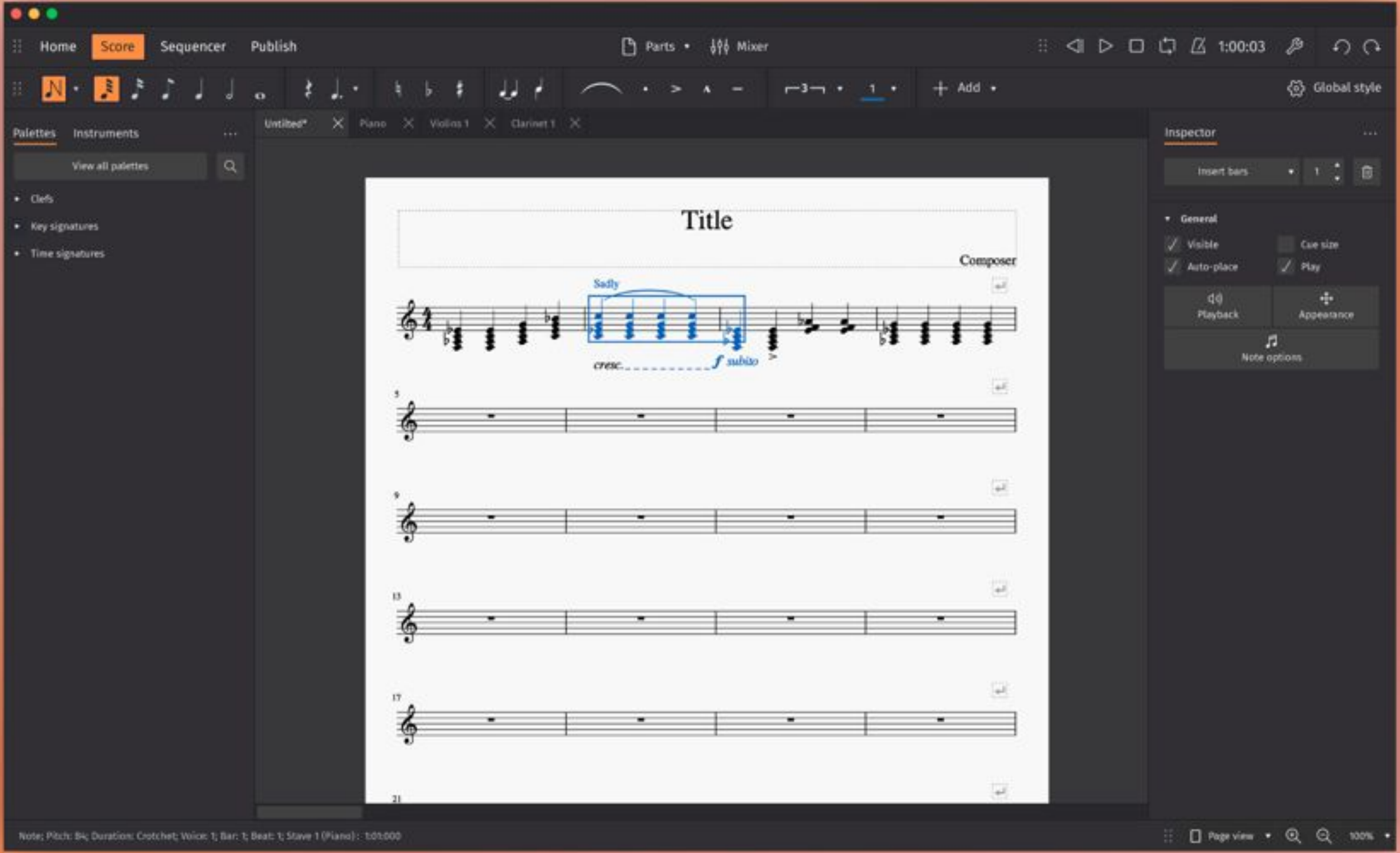
Here's a description of the extension. Keep it short and sweet.

Early Music

Here's a description of the extension. Keep it short and sweet.



Note; Pitch: B4; Duration: Crotchet; Voice: 1; Bar: 1; Beat: 1; Stave 1 (Piano) : 1:01:000

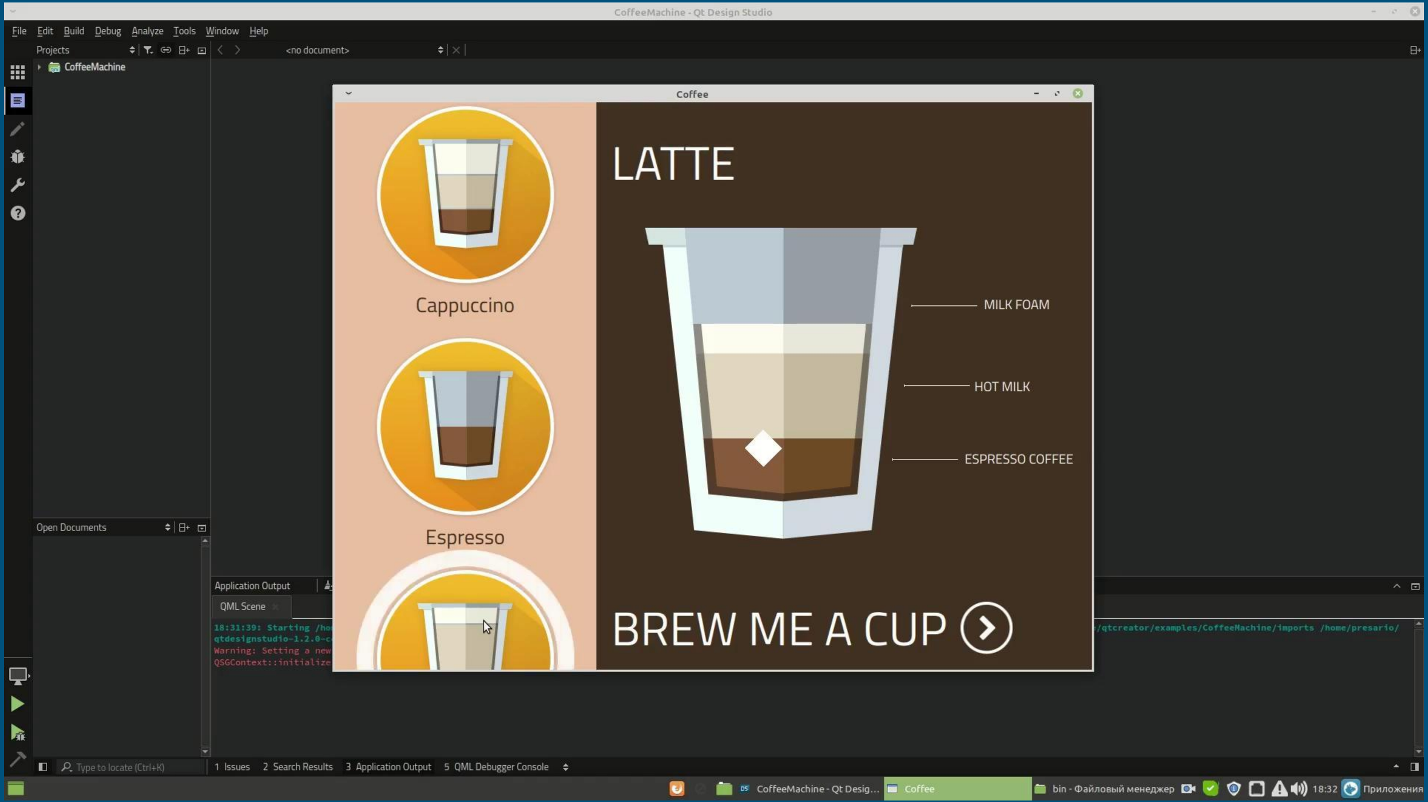


Note; Pitch: B4; Duration: Crotchet; Voice: 1; Bar: 1; Beat: 1; Stave 1 (Piano) : 1:01:000

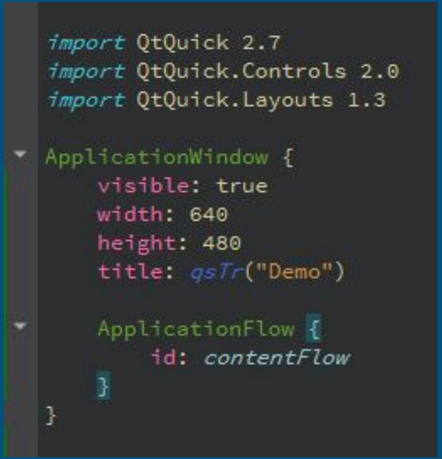
THE BASIC STRATEGIES

- FULL QML (100%)
- Mix of QML + QWidgets (50 - 99%)
- Mix of QWidgets + QML (10 - 40%)

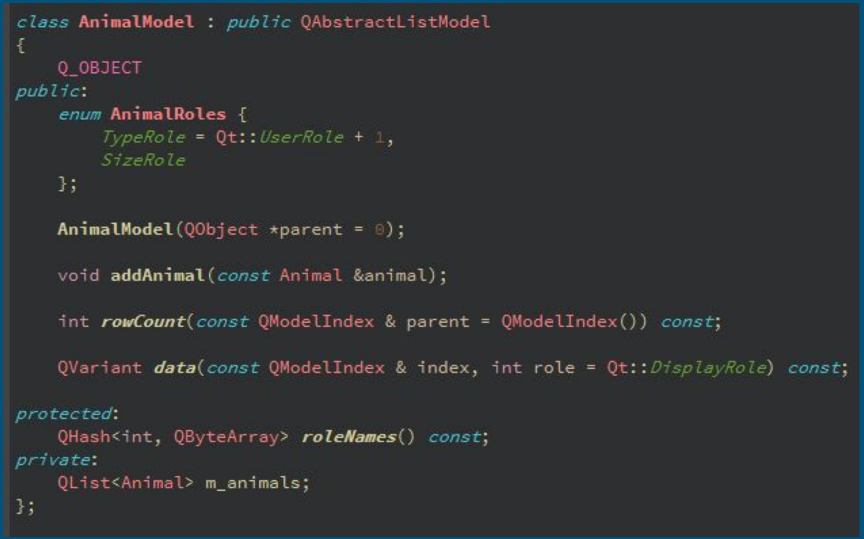
FULL QML STRATEGY



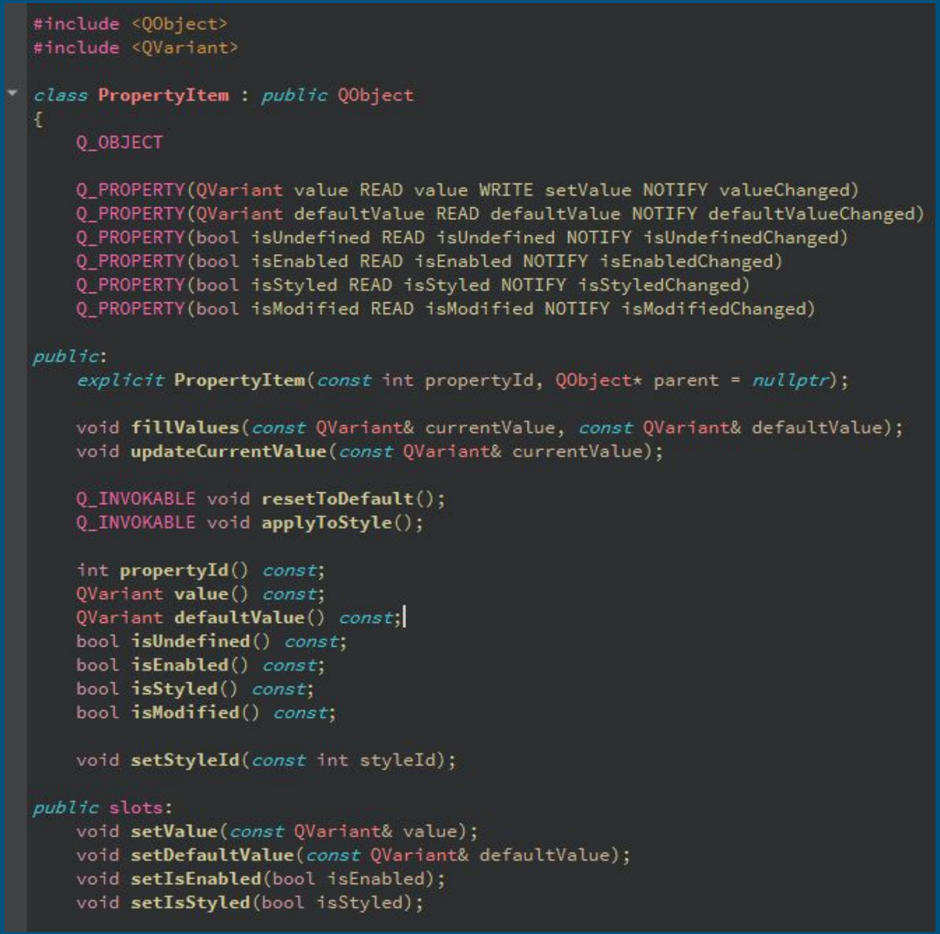
```
18:31:39: Starting /ho                    /qtcreator/examples/CoffeeMachine/imports /home/presario/
qtdesignstudio-1.2.0-c
Warning: Setting a new
QSGContext::initialize
```



```
import QtQuick 2.7
import QtQuick.Controls 2.0
import QtQuick.Layouts 1.3

ApplicationWindow {
    visible: true
    width: 640
    height: 480
    title: qsTr("Demo")

    ApplicationFlow {
        id: contentFlow
    }
}
```



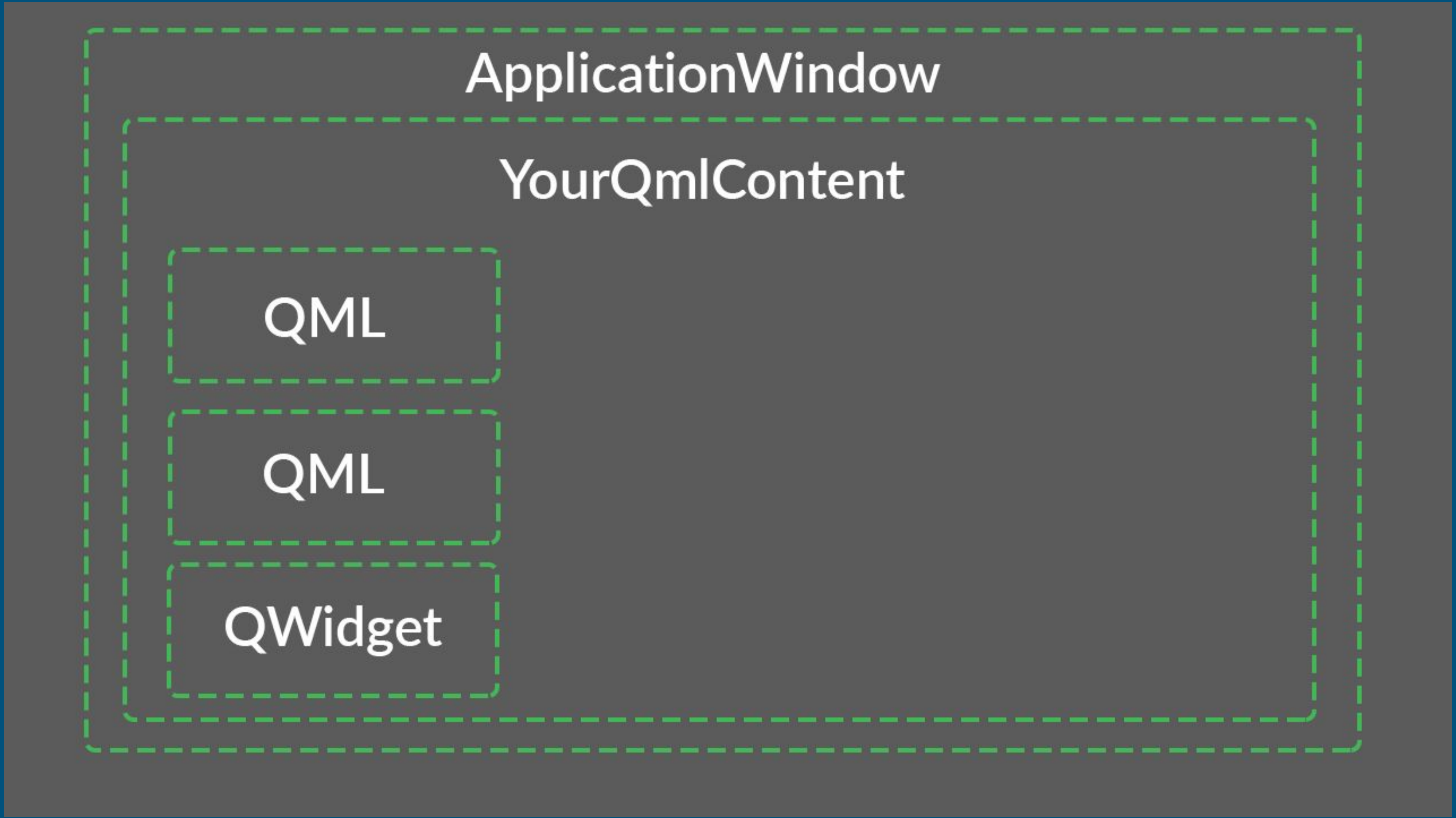
```
class AnimalModel : public QAbstractListModel
{
    Q_OBJECT
public:
    enum AnimalRoles {
        TypeRole = Qt::UserRole + 1,
        SizeRole
    };

    AnimalModel(QObject *parent = 0);

    void addAnimal(const Animal &animal);

    int rowCount(const QModelIndex & parent = QModelIndex()) const;

    QVariant data(const QModelIndex & index, int role = Qt::DisplayRole) const;

protected:
    QHash<int, QByteArray> roleNames() const;
private:
    QList<Animal> m_animals;
};
```

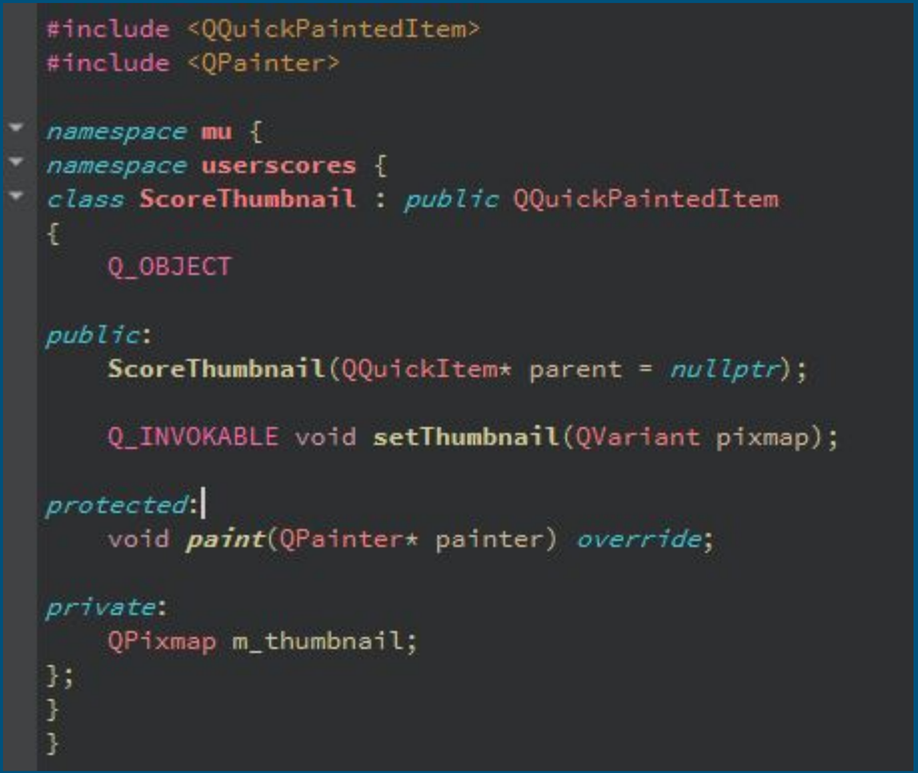


```
#include <QObject>
#include <QVariant>

class PropertyItem : public QObject
{
    Q_OBJECT

    Q_PROPERTY(QVariant value READ value WRITE setValue NOTIFY valueChanged)
    Q_PROPERTY(QVariant defaultValue READ defaultValue NOTIFY defaultValueChanged)
    Q_PROPERTY(bool isUndefined READ isUndefined NOTIFY isUndefinedChanged)
    Q_PROPERTY(bool isEnabled READ isEnabled NOTIFY isEnabledChanged)
    Q_PROPERTY(bool isStyled READ isStyled NOTIFY isStyledChanged)
    Q_PROPERTY(bool isModified READ isModified NOTIFY isModifiedChanged)

public:
    explicit PropertyItem(const int propertyId, QObject* parent = nullptr);

    void fillValues(const QVariant& currentValue, const QVariant& defaultValue);
    void updateCurrentValue(const QVariant& currentValue);

    Q_INVOKABLE void resetToDefault();
    Q_INVOKABLE void applyToStyle();

    int propertyId() const;
    QVariant value() const;
    QVariant defaultValue() const;
    bool isUndefined() const;
    bool isEnabled() const;
    bool isStyled() const;
    bool isModified() const;

    void setStyleId(const int styleId);

public slots:
    void setValue(const QVariant& value);
    void setDefaultValue(const QVariant& defaultValue);
    void setIsEnabled(bool isEnabled);
    void setIsStyled(bool isStyled);
```

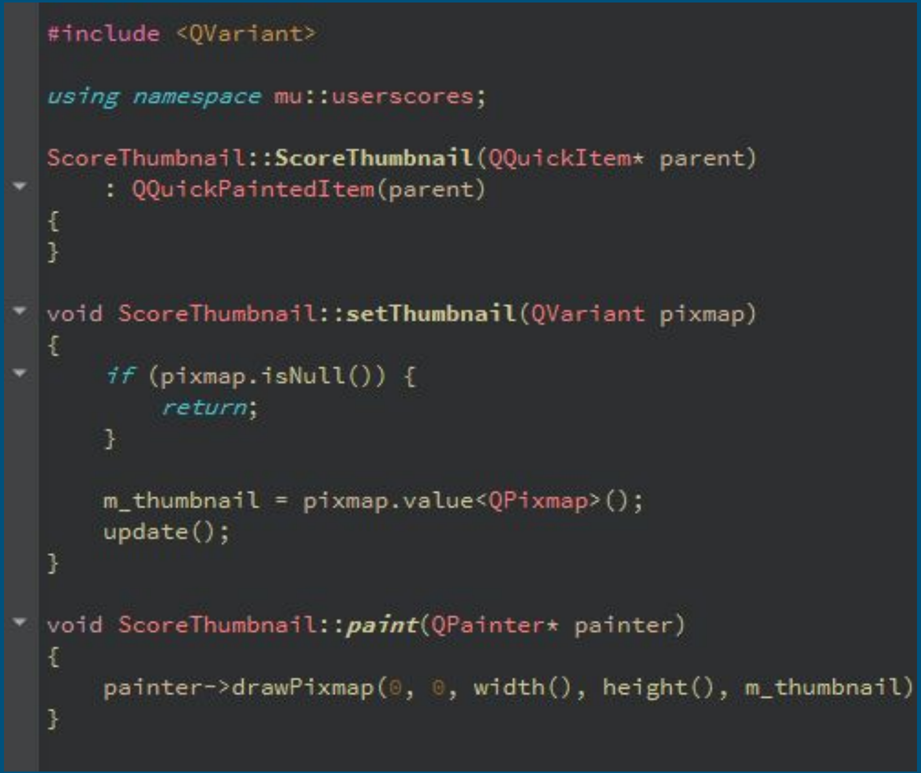
qmlRegisterType<YourType>(..., ...)

MIX OF QML + QWIDGETS

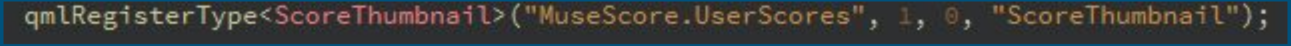




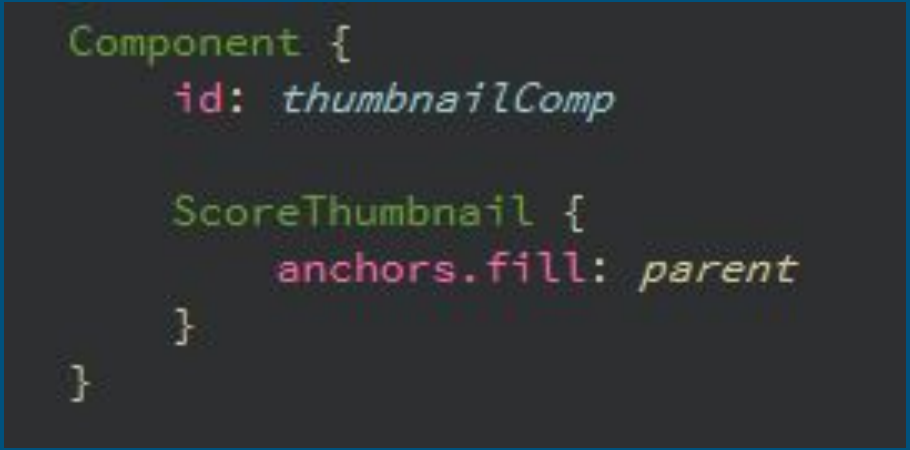
```cpp
#include <QQuickPaintedItem>
#include <QPainter>

namespace mu {
namespace userscores {
class ScoreThumbnail : public QQuickPaintedItem
{
    Q_OBJECT

public:
    ScoreThumbnail(QQuickItem* parent = nullptr);

    Q_INVOKABLE void setThumbnail(QVariant pixmap);

protected:
    void paint(QPainter* painter) override;

private:
    QPixmap m_thumbnail;
};
}
}
```



```cpp
#include <QVariant>

using namespace mu::userscores;

ScoreThumbnail::ScoreThumbnail(QQuickItem* parent)
    : QQuickPaintedItem(parent)
{
}

void ScoreThumbnail::setThumbnail(QVariant pixmap)
{
    if (pixmap.isNull()) {
        return;
    }

    m_thumbnail = pixmap.value<QPixmap>();
    update();
}

void ScoreThumbnail::paint(QPainter* painter)
{
    painter->drawPixmap(0, 0, width(), height(), m_thumbnail)
}
```

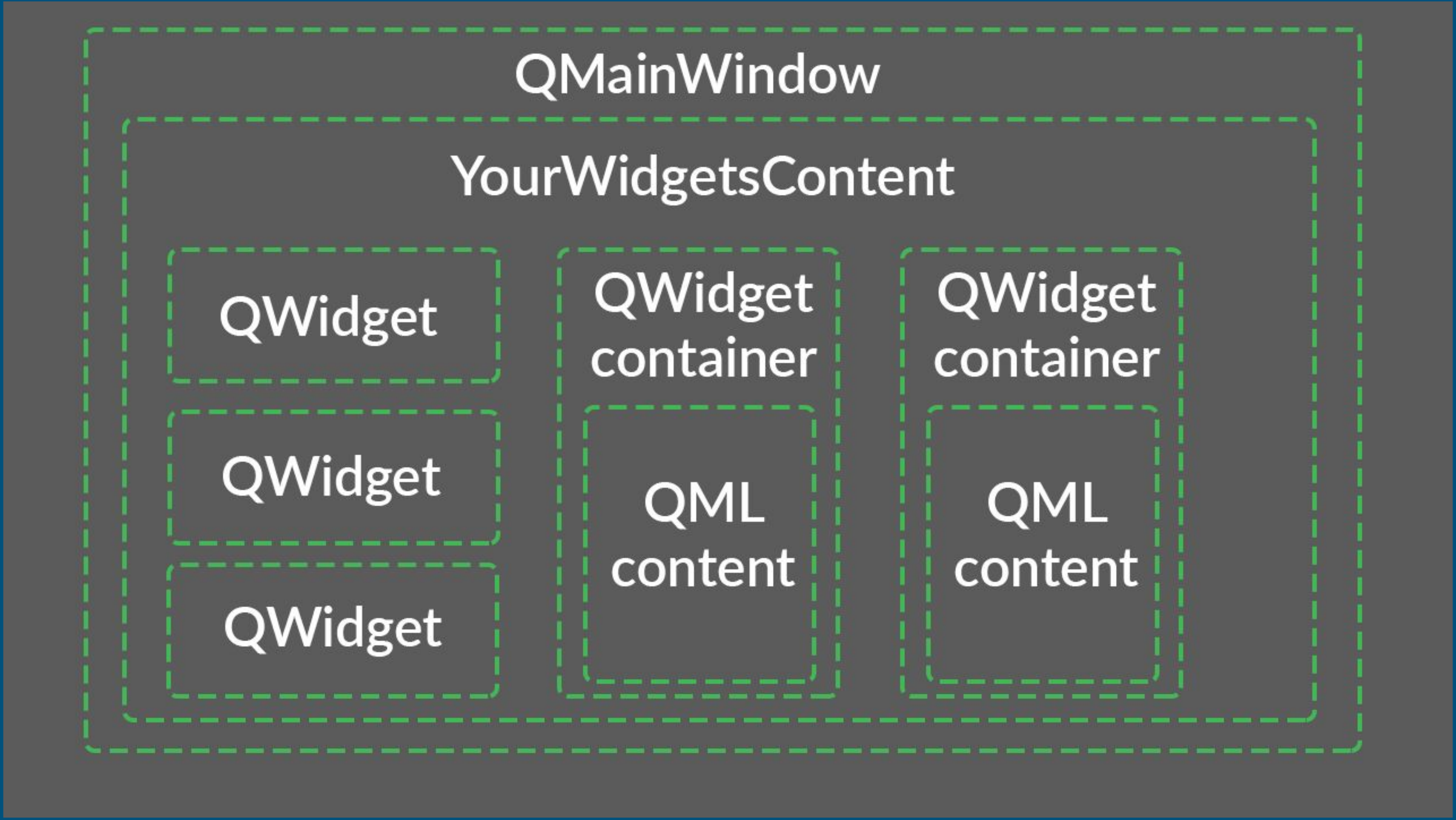


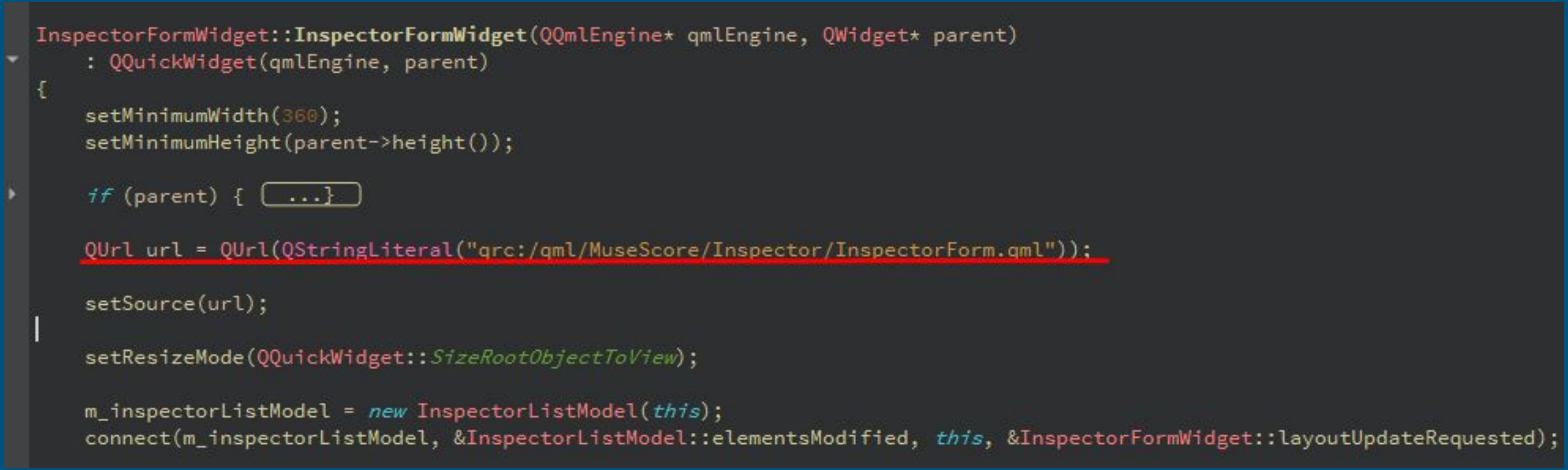
```
qmlRegisterType<ScoreThumbnail>("MuseScore.UserScores", 1, 0, "ScoreThumbnail");
```



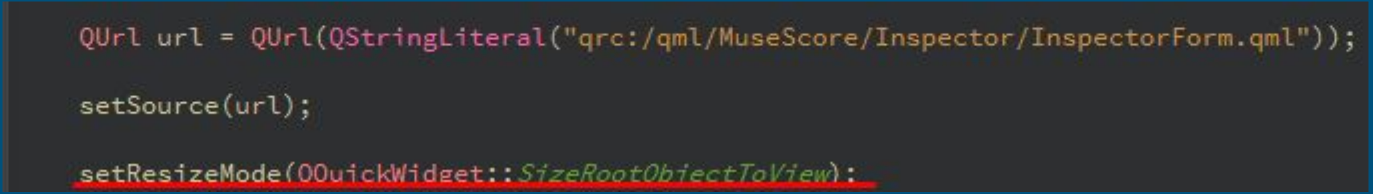
```
Component {
    id: thumbnailComp

    ScoreThumbnail {
        anchors.fill: parent
    }
}
```

MIX OF QWIDGETS + QML

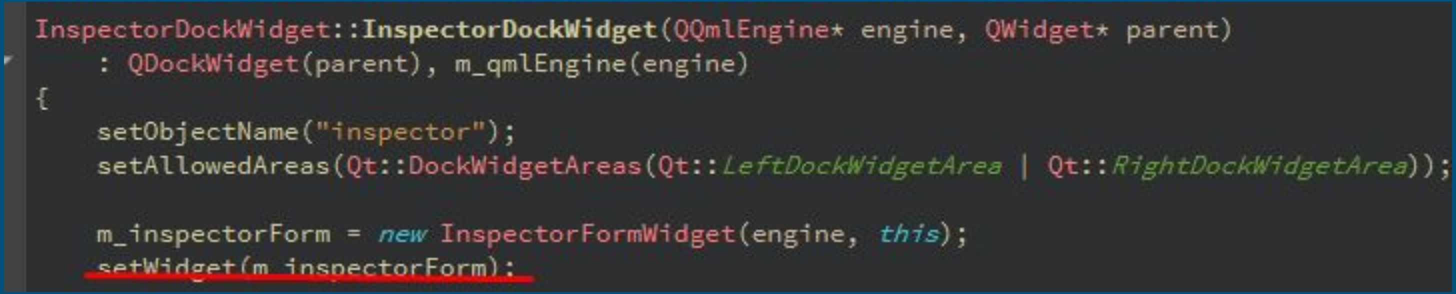




```
InspectorFormWidget::InspectorFormWidget(QQmlEngine* qmlEngine, QWidget* parent)
    : QQuickWidget(qmlEngine, parent)
{
    setMinimumWidth(360);
    setMinimumHeight(parent->height());

    if (parent) { ...}

    QUrl url = QUrl(QStringLiteral("qrc:/qml/MuseScore/Inspector/InspectorForm.qml"));

    setSource(url);

    setResizeMode(QQuickWidget::SizeRootObjectToView);

    m_inspectorListModel = new InspectorListModel(this);
    connect(m_inspectorListModel, &InspectorListModel::elementsModified, this, &InspectorFormWidget::layoutUpdateRequested);
```



```
    QUrl url = QUrl(QStringLiteral("qrc:/qml/MuseScore/Inspector/InspectorForm.qml"));

    setSource(url);

    setResizeMode(QQuickWidget::SizeRootObjectToView);
```

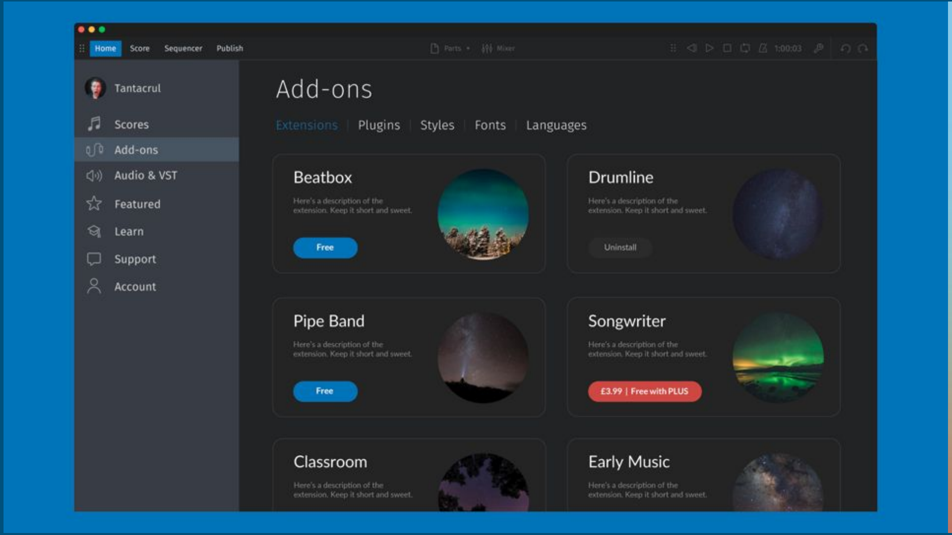


```
InspectorDockWidget::InspectorDockWidget(QQmlEngine* engine, QWidget* parent)
    : QDockWidget(parent), m_qmlEngine(engine)
{
    setObjectName("inspector");
    setAllowedAreas(Qt::DockWidgetAreas(Qt::LeftDockWidgetArea | Qt::RightDockWidgetArea));

    m_inspectorForm = new InspectorFormWidget(engine, this);
    setWidget(m_inspectorForm);
```

MuseScore editor example

- Do we need a QML?

- YES



Add-ons

Beatbox

Here's a description of the extension. Keep it short and sweet.

Drumline

Here's a description of the extension. Keep it short and sweet.

Pipe Band

Here's a description of the extension. Keep it short and sweet.

Songwriter

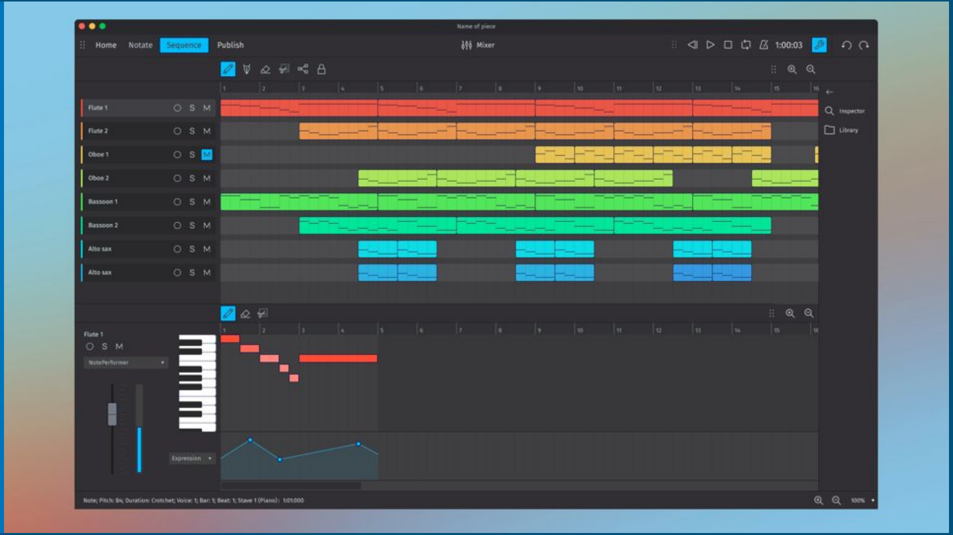
Here's a description of the extension. Keep it short and sweet.

Classroom

Here's a description of the extension. Keep it short and sweet.

Early Music

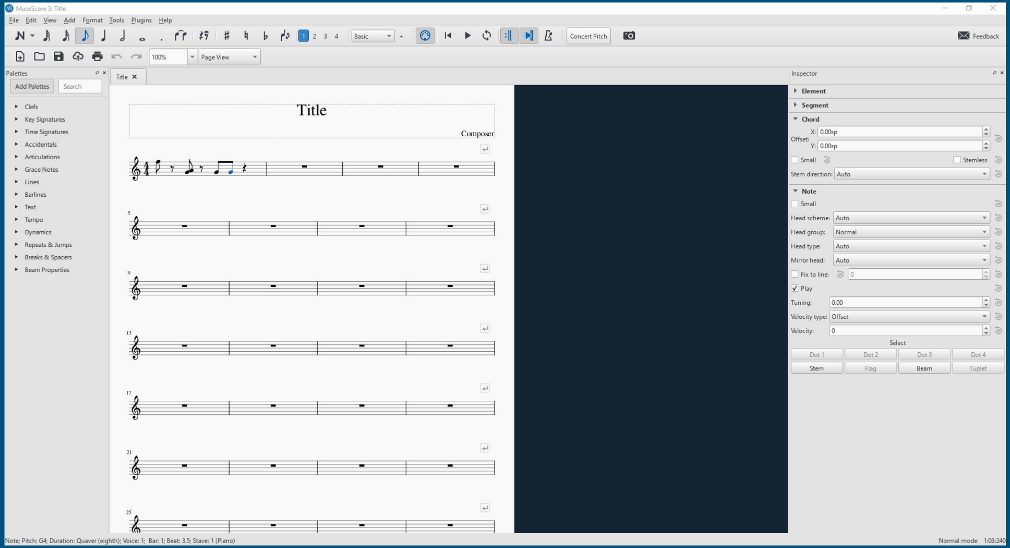
Here's a description of the extension. Keep it short and sweet.



Note; Pitch: 84; Duration: Crotchet; Voice: 1; Bar: 1; Beat: 1; Stave 1 (Piano) : 1:01:000

MuseScore editor example

- Are we building a project from scratch?
  - NO



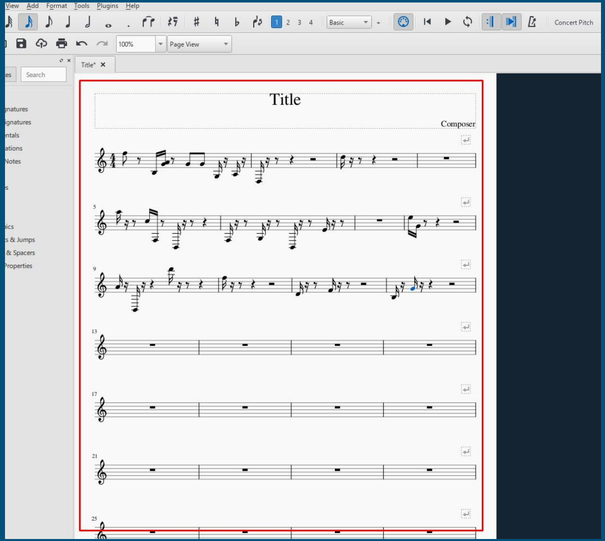
Offset: X: 0.00sp
Y: 0.00sp
□ Small □ Stemless
Stem direction: Auto
□ Small
Head scheme: Auto
Head group: Normal
Head type: Auto
Mirror head: Auto
□ Fix to line: 0
☑ Play
Tuning: 0.00
Velocity type: Offset
Velocity: 0

Note; Pitch: G4; Duration: Quaver (eighth); Voice: 1; Bar: 1; Beat: 3.5; Stave: 1 (Piano)

Normal mode 1:03:240

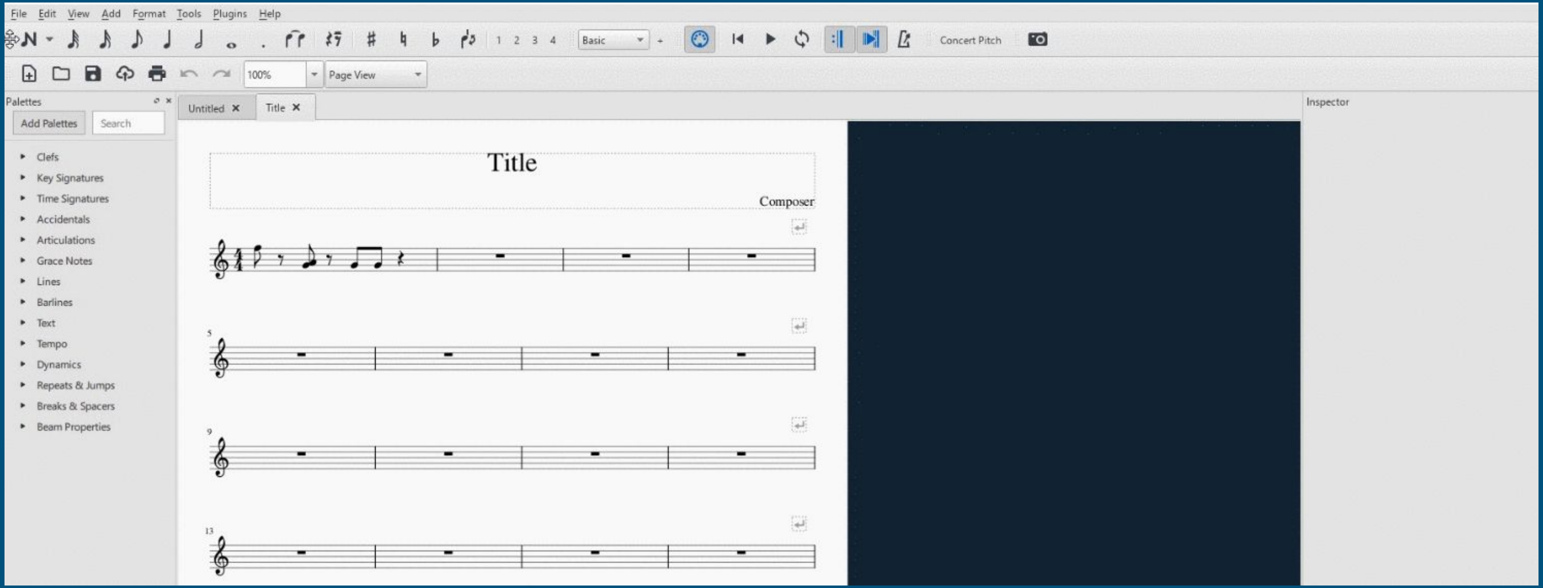
MuseScore editor example

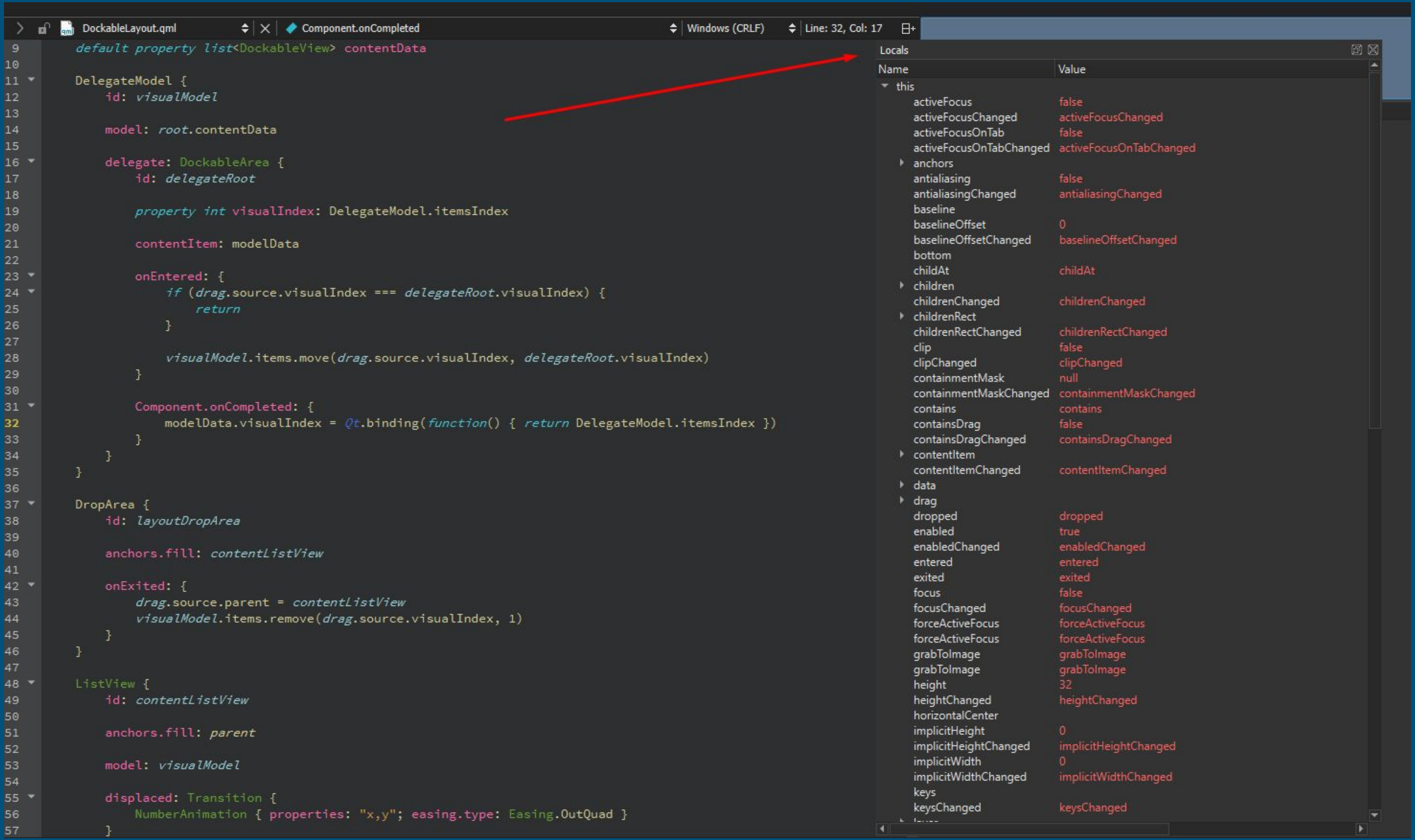
- Do we have things that we would like to keep on QWidgets?
  - YES



MuseScore editor example

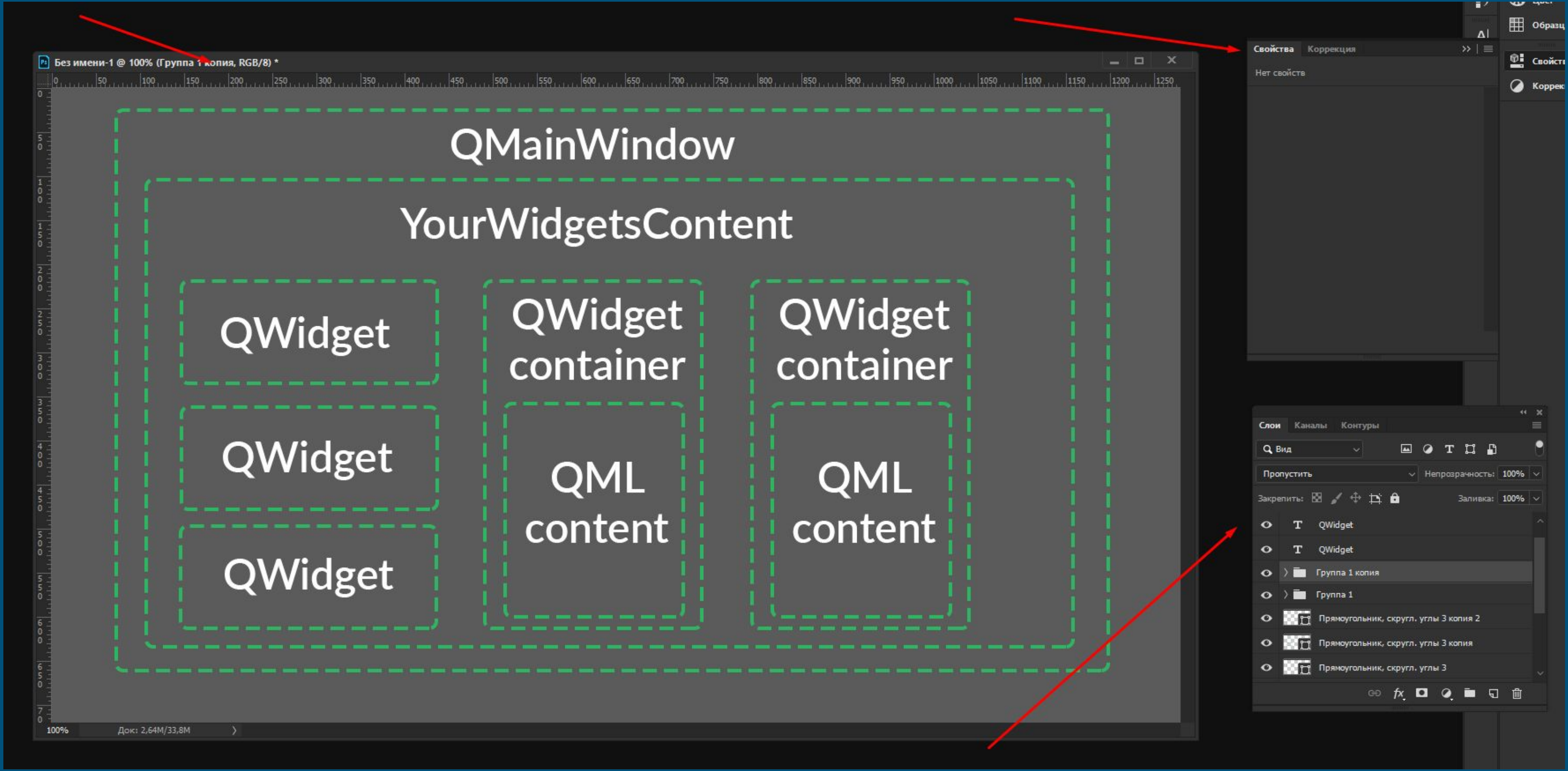
- Are there things we need to keep in design that are nearly impossible to implement in QtQuick?
    - YES





```qml
    default property list<DockableView> contentData

    DelegateModel {
        id: visualModel

        model: root.contentData

        delegate: DockableArea {
            id: delegateRoot

            property int visualIndex: DelegateModel.itemsIndex

            contentItem: modelData

            onEntered: {
                if (drag.source.visualIndex === delegateRoot.visualIndex) {
                    return
                }

                visualModel.items.move(drag.source.visualIndex, delegateRoot.visualIndex)
            }

            Component.onCompleted: {
                modelData.visualIndex = Qt.binding(function() { return DelegateModel.itemsIndex })
            }
        }
    }

    DropArea {
        id: layoutDropArea

        anchors.fill: contentListView

        onExited: {
            drag.source.parent = contentListView
            visualModel.items.remove(drag.source.visualIndex, 1)
        }
    }

    ListView {
        id: contentListView

        anchors.fill: parent

        model: visualModel

        displaced: Transition {
            NumberAnimation { properties: "x,y"; easing.type: Easing.OutQuad }
        }
```

```
Name                        Value
▾ this
    activeFocus               false
    activeFocusChanged        activeFocusChanged
    activeFocusOnTab          false
    activeFocusOnTabChanged   activeFocusOnTabChanged
  ▸ anchors
    antialiasing              false
    antialiasingChanged       antialiasingChanged
    baseline
    baselineOffset            0
    baselineOffsetChanged     baselineOffsetChanged
    bottom
    childAt                   childAt
  ▸ children
    childrenChanged           childrenChanged
  ▸ childrenRect
    childrenRectChanged       childrenRectChanged
    clip                      false
    clipChanged               clipChanged
    containmentMask           null
    containmentMaskChanged    containmentMaskChanged
    contains                  contains
    containsDrag              false
    containsDragChanged       containsDragChanged
  ▸ contentItem
    contentItemChanged        contentItemChanged
  ▸ data
  ▸ drag
    dropped                   dropped
    enabled                   true
    enabledChanged            enabledChanged
    entered                   entered
    exited                    exited
    focus                     false
    focusChanged              focusChanged
    forceActiveFocus          forceActiveFocus
    forceActiveFocus          forceActiveFocus
    grabToImage               grabToImage
    grabToImage               grabToImage
    height                    32
    heightChanged             heightChanged
    horizontalCenter
    implicitHeight            0
    implicitHeightChanged     implicitHeightChanged
    implicitWidth             0
    implicitWidthChanged      implicitWidthChanged
    keys
    keysChanged               keysChanged
```



Без имени-1 @ 100% (Группа 1 копия, RGB/8) *

Свойства Коррекция

Нет свойств

Образц

Свойств

Коррек

Слои Каналы Контуры

Вид

Пропустить

Непрозрачность: 100%

Закрепить:

Заливка: 100%

QWidget

QWidget

Группа 1 копия

Группа 1

Прямоугольник, скругл. углы 3 копия 2

Прямоугольник, скругл. углы 3 копия

Прямоугольник, скругл. углы 3

100%

Док: 2,64M/33,8M

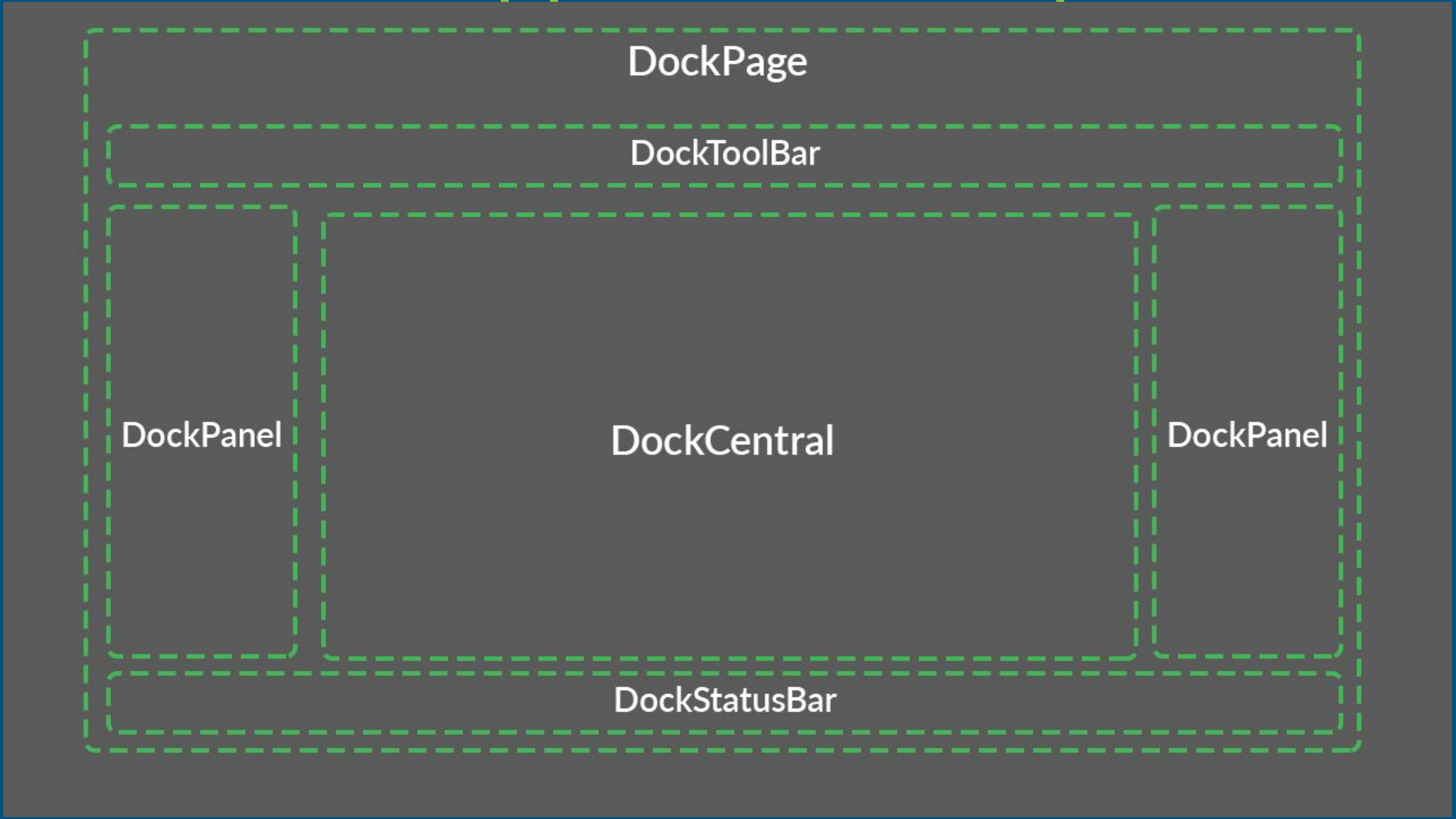
THE PITFALLS

1. DOCKABLE SYSTEM

- Implement your mechanism using QDockWidgets in QML

- Wait for the 1.1 release of the impressive KDDockWidgets library with QtQuick support from our friends at KDAB

Our approach example



DockPage

DockToolBar

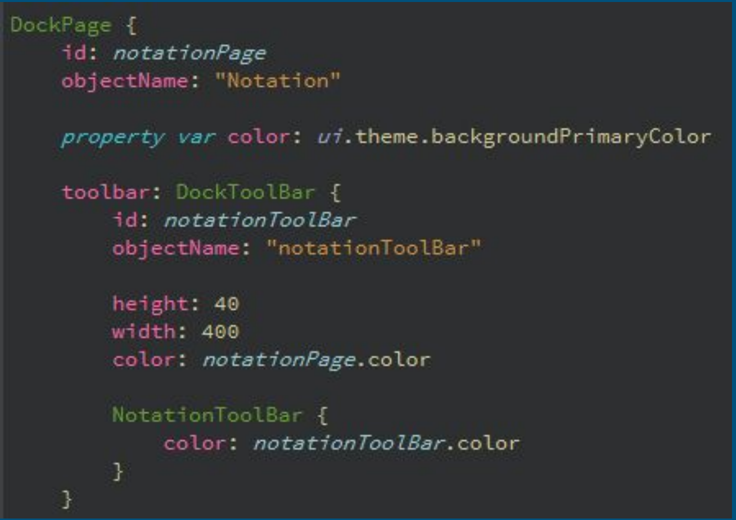
DockPanel

DockCentral

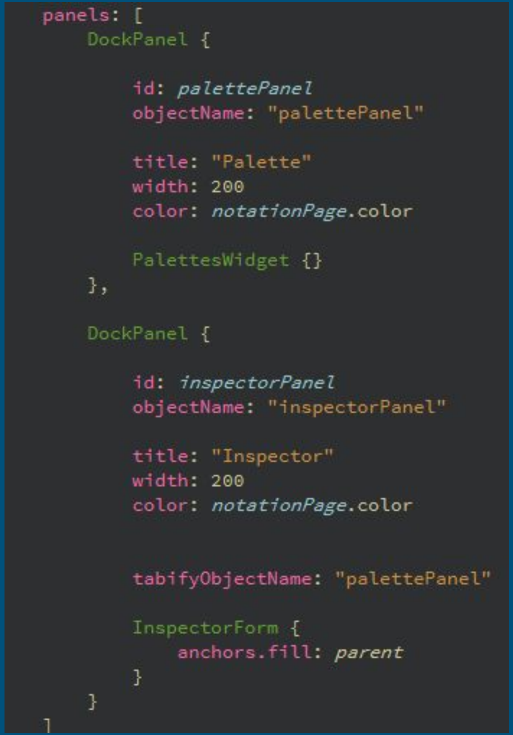
DockPanel

DockStatusBar

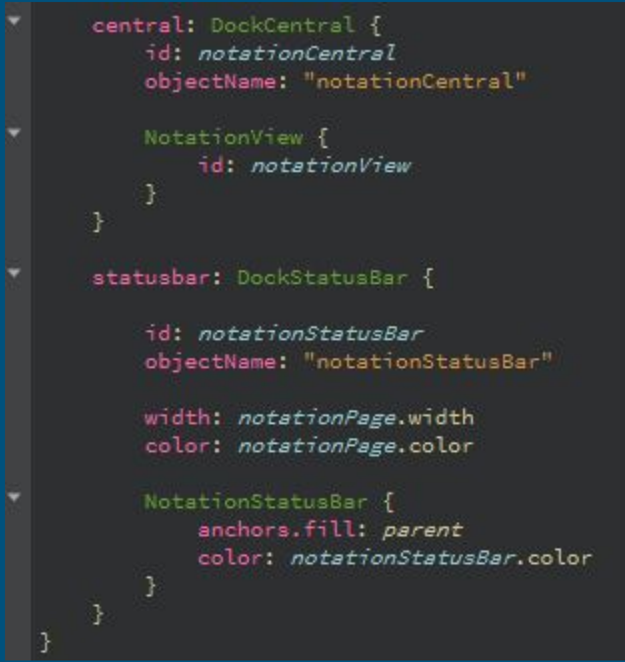
Our approach example



```
DockPage {
    id: notationPage
    objectName: "Notation"

    property var color: ui.theme.backgroundPrimaryColor

    toolbar: DockToolBar {
        id: notationToolBar
        objectName: "notationToolBar"

        height: 40
        width: 400
        color: notationPage.color

        NotationToolBar {
            color: notationToolBar.color
        }
    }
```



```
panels: [
    DockPanel {

        id: palettePanel
        objectName: "palettePanel"

        title: "Palette"
        width: 200
        color: notationPage.color

        PalettesWidget {}
    },

    DockPanel {

        id: inspectorPanel
        objectName: "inspectorPanel"

        title: "Inspector"
        width: 200
        color: notationPage.color


        tabifyObjectName: "palettePanel"

        InspectorForm {
            anchors.fill: parent
        }
    }
]
```



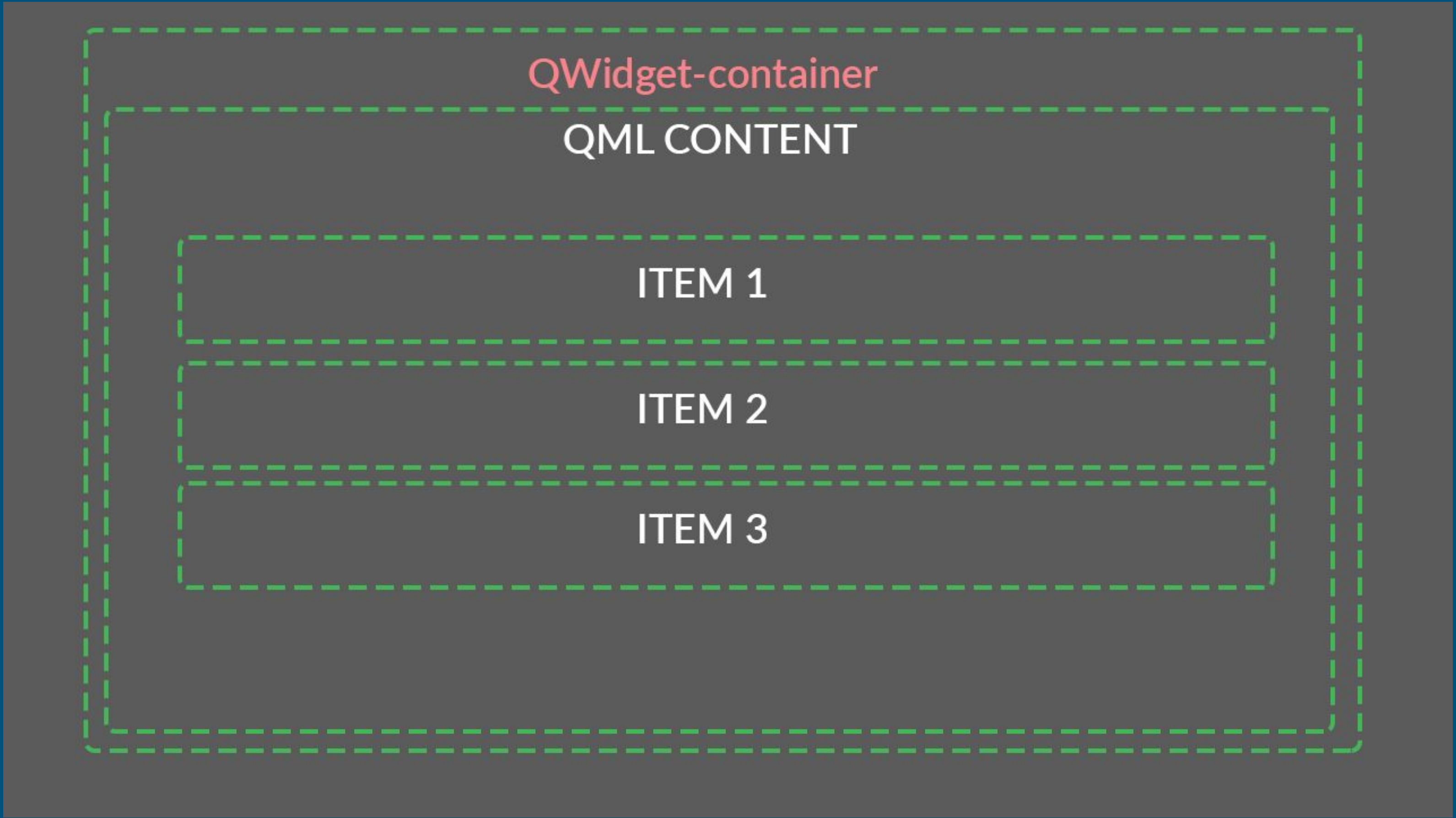
```
    central: DockCentral {
        id: notationCentral
        objectName: "notationCentral"

        NotationView {
            id: notationView
        }
    }

    statusbar: DockStatusBar {

        id: notationStatusBar
        objectName: "notationStatusBar"

        width: notationPage.width
        color: notationPage.color

        NotationStatusBar {
            anchors.fill: parent
            color: notationStatusBar.color
        }
    }
}
```

THE PITFALLS

2. Focus handling

- Cyclic transfer of focus within one container

- *Pressed TAB key*



QWidget-container

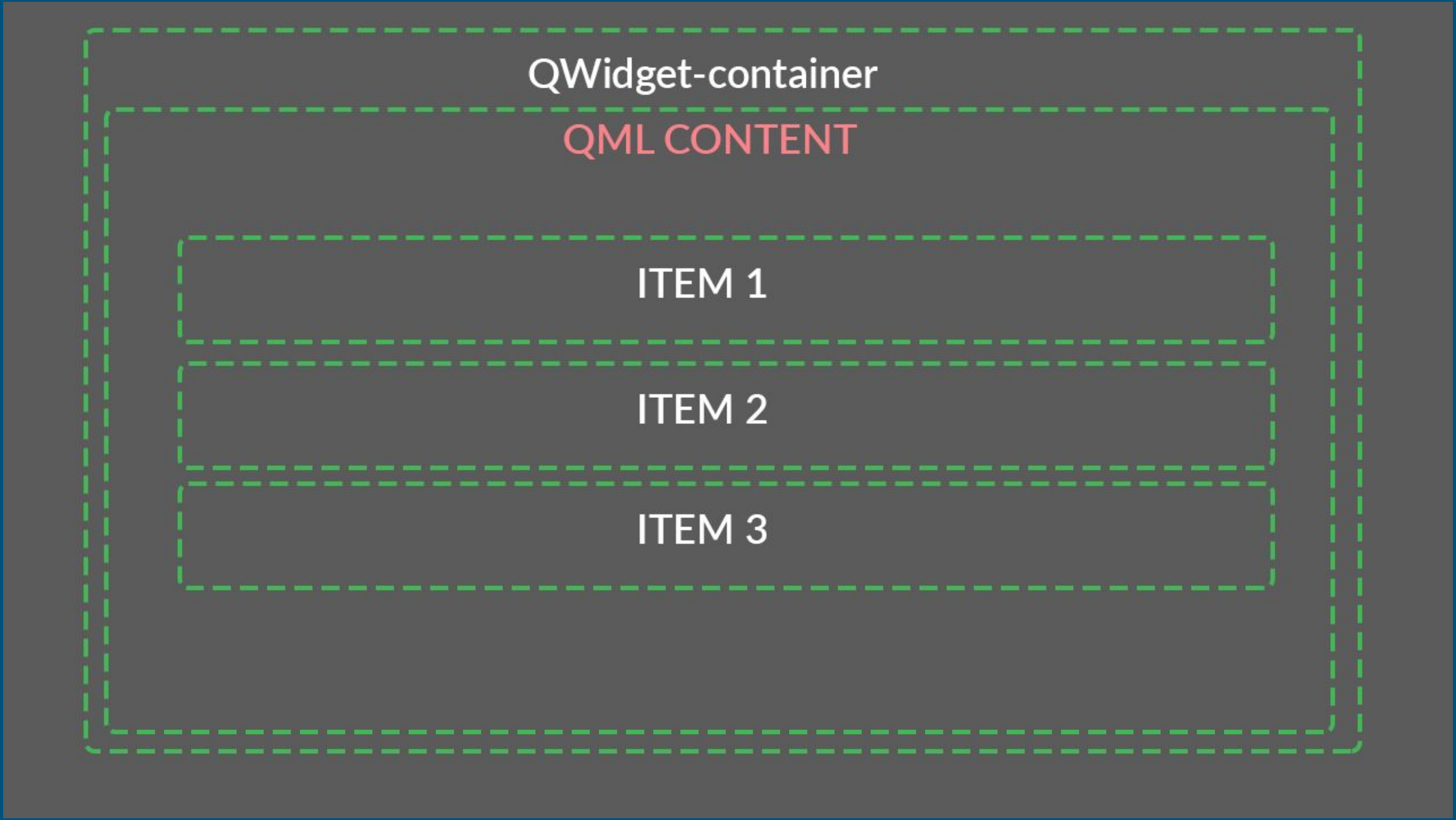
QML CONTENT

ITEM 1

ITEM 2

ITEM 3

- *Pressed TAB key*



QWidget-container

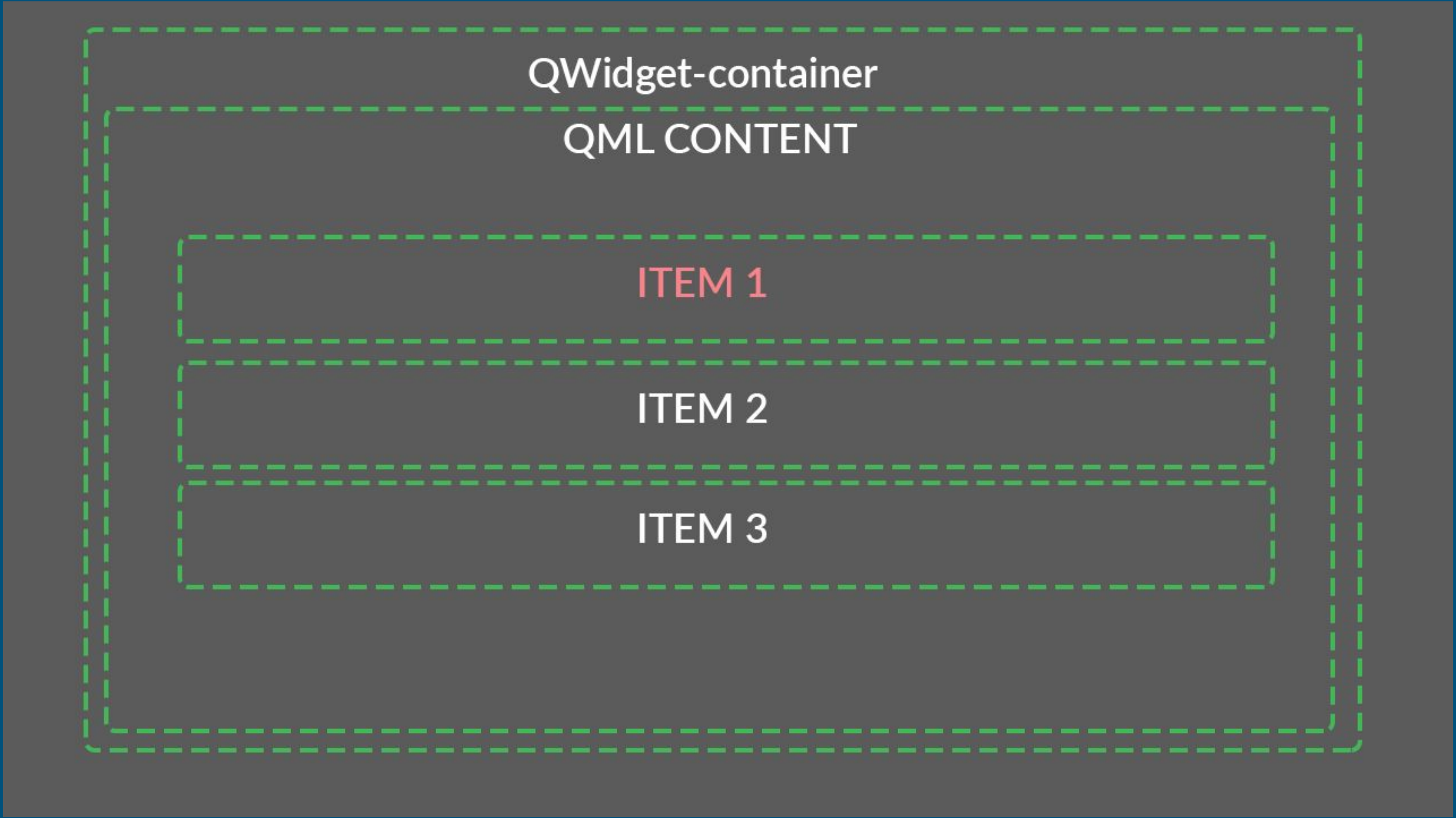
QML CONTENT

ITEM 1

ITEM 2

ITEM 3

- *Pressed TAB key*



QWidget-container

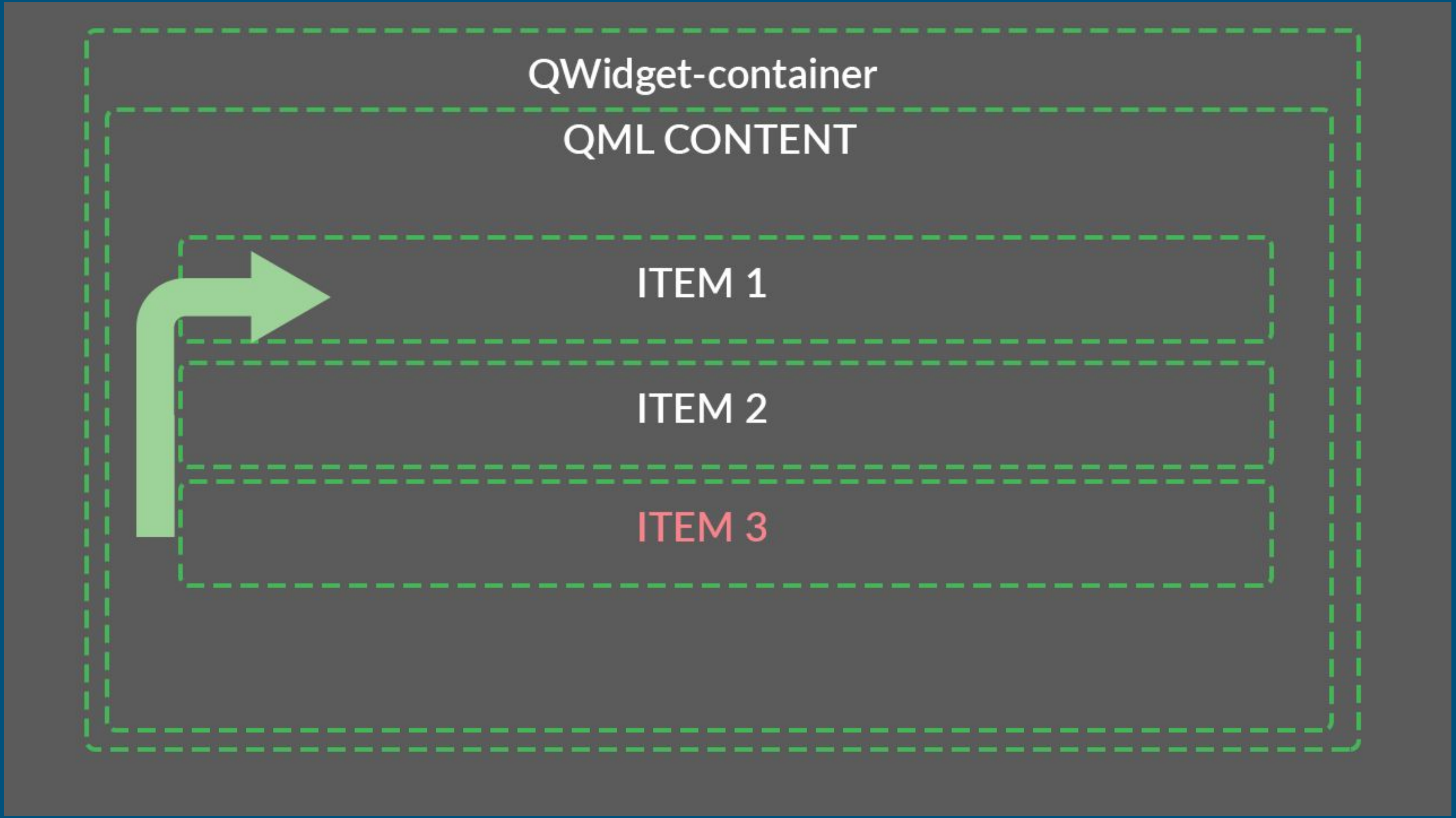
QML CONTENT

ITEM 1

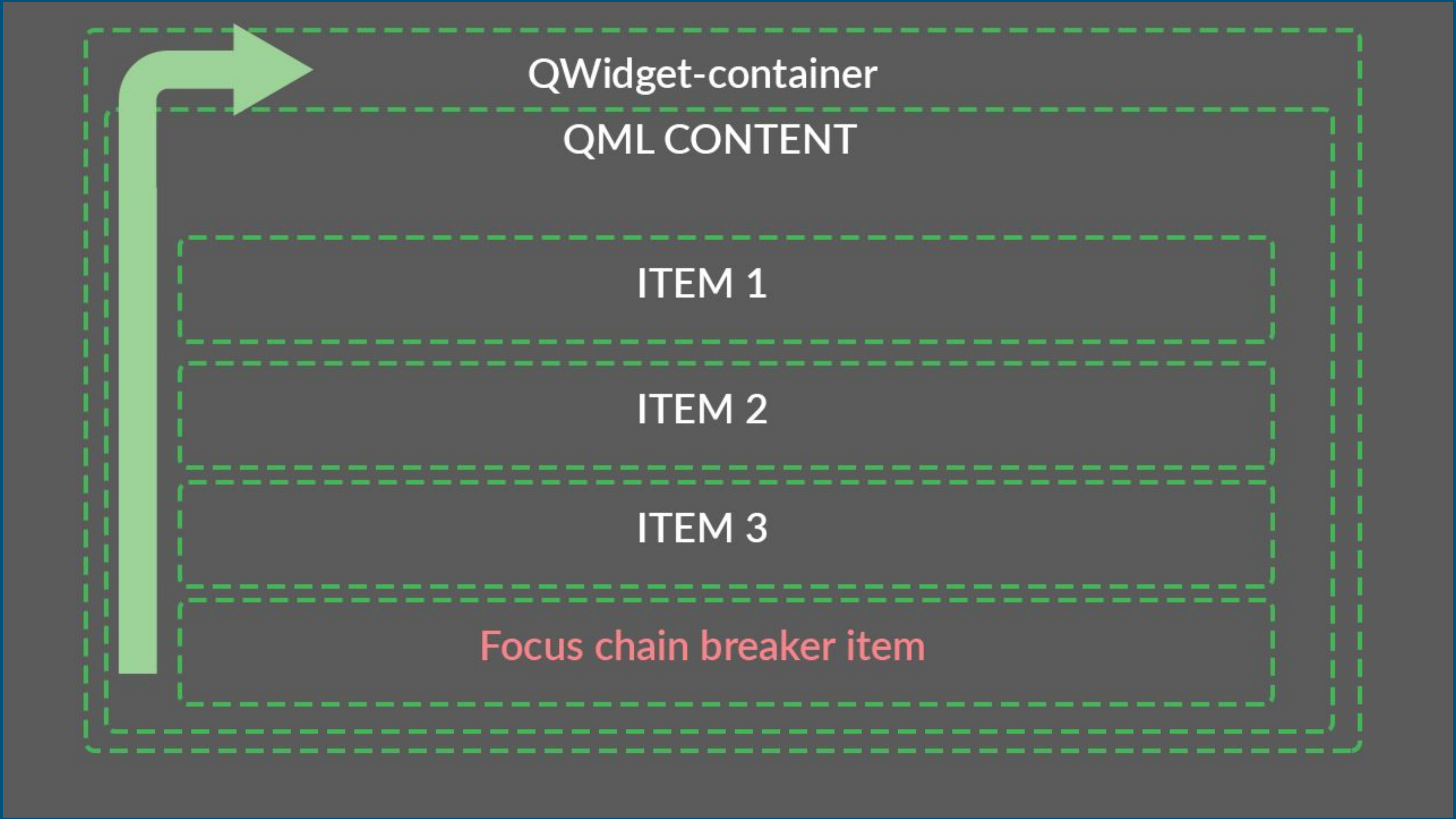
ITEM 2

ITEM 3

- *Pressed TAB key*



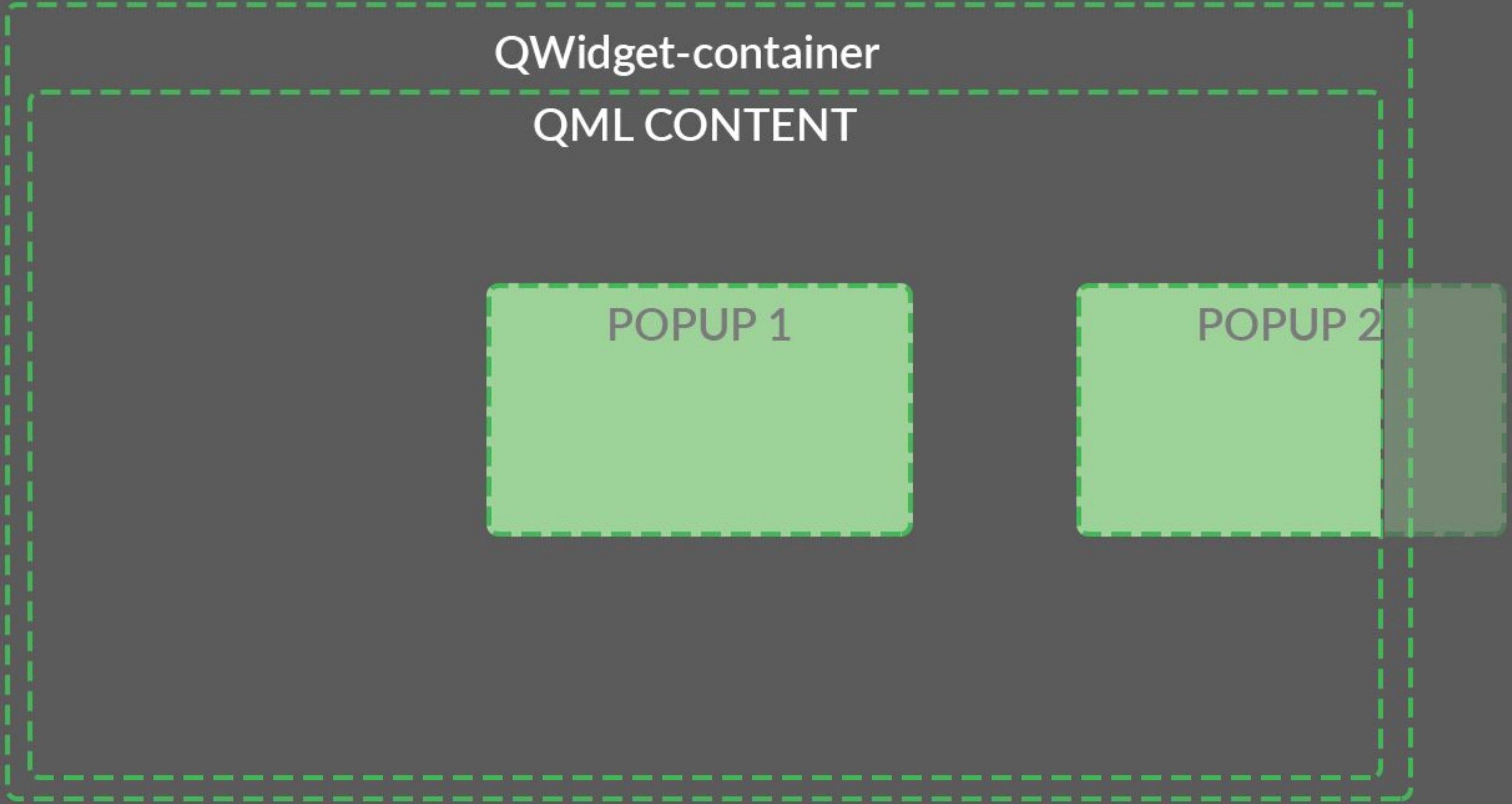
- *Pressed TAB key*



THE PITFALLS

3. Popups overlapping

- QML content cannot be rendered outside the QWidget container it belongs to



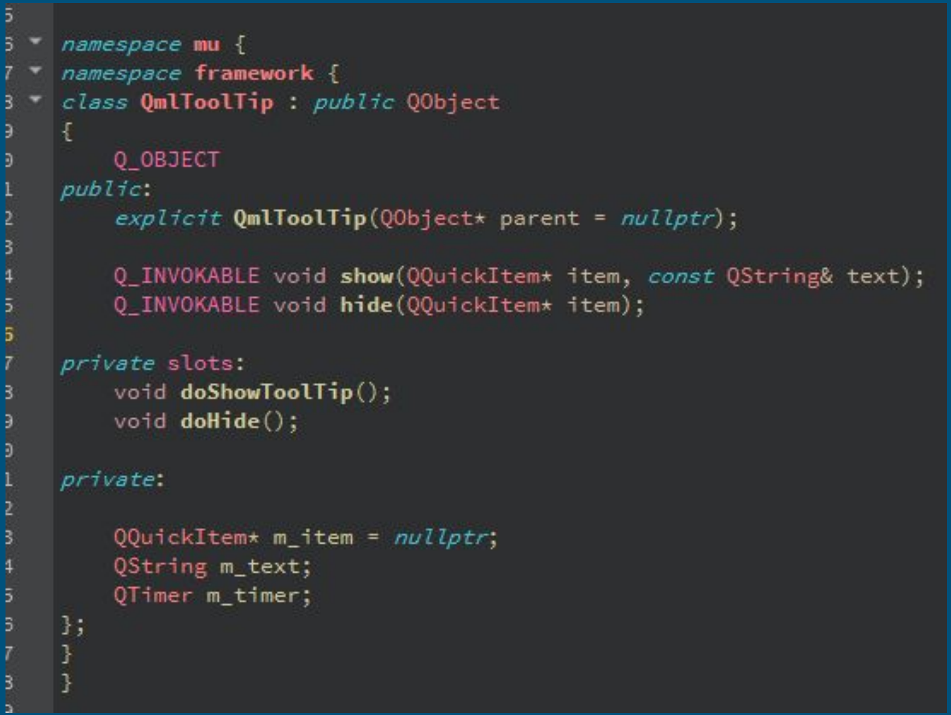
QWidget-container

QML CONTENT

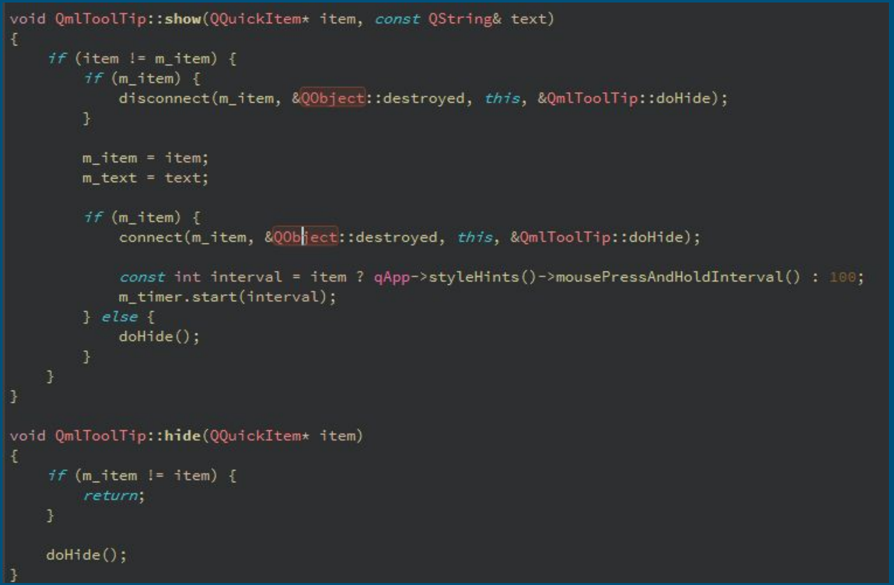
POPUP 1

POPUP 2

ToolTip example



```cpp
namespace mu {
namespace framework {
class QmlToolTip : public QObject
{
    Q_OBJECT
public:
    explicit QmlToolTip(QObject* parent = nullptr);

    Q_INVOKABLE void show(QQuickItem* item, const QString& text);
    Q_INVOKABLE void hide(QQuickItem* item);

private slots:
    void doShowToolTip();
    void doHide();

private:

    QQuickItem* m_item = nullptr;
    QString m_text;
    QTimer m_timer;
};
}
}
```



```cpp
void QmlToolTip::show(QQuickItem* item, const QString& text)
{
    if (item != m_item) {
        if (m_item) {
            disconnect(m_item, &QObject::destroyed, this, &QmlToolTip::doHide);
        }

        m_item = item;
        m_text = text;

        if (m_item) {
            connect(m_item, &QObject::destroyed, this, &QmlToolTip::doHide);

            const int interval = item ? qApp->styleHints()->mousePressAndHoldInterval() : 100;
            m_timer.start(interval);
        } else {
            doHide();
        }
    }
}

void QmlToolTip::hide(QQuickItem* item)
{
    if (m_item != item) {
        return;
    }

    doHide();
}
```

THANKS!