Construindo Aplicativos com QML

Enoque Joseneas

O que é QML?

- QML = ***Q**t **M**eta-object **L**anguage*
- É uma linguagem declarativa para desenvolver *User Interfaces*
- Tem foco em agilidade, facilidade e reuso de componentes
- Suporte a código JavaScript Inline ou via arquivo *.js* externo
- Interpretada por um Qt framework
- Possui fácil interação com c++
- Suporte a rede (http) através de javascript ou via Qt
- Suporte a banco de dados via javascript ou Qt
- *"QML é um um tipo HTML + CSS da Web"* Só que mais poderosa!
- QML + Qt/C++ = ***Qt Quick***
- Excelente documentação com exemplos

O que é QML?

- **Tem suporte a sensores**
  - O módulo ***Qt Sensors*** permite que aos aplicativos ler informações provenientes de sensores como acelerômetros e sensores de inclinação. Existe uma API comum QML para diferentes plataformas e pode ser estendido em C ++.
- **Conteúdo Multimídia**
  - O módulo ***Qt Multimedia*** permite aos aplicativos interagir com conteúdos multimídia através de um conjunto conveniente de tipos QML. Estes tipos QML pode ser estendido em C ++.
    - Módulos para Áudio
    - Câmera
    - Vídeo

O que é QML?

- **Descreve uma árvore de objetos e propriedades**

```
Item {
    propertyA: 100
    propertyB: 100

    children: [
        Item { propertyC: 100 },
        Item { propertyC: 200 }
    ]
}
```

O que é QML?

- **Permite incorporar código Javascript**

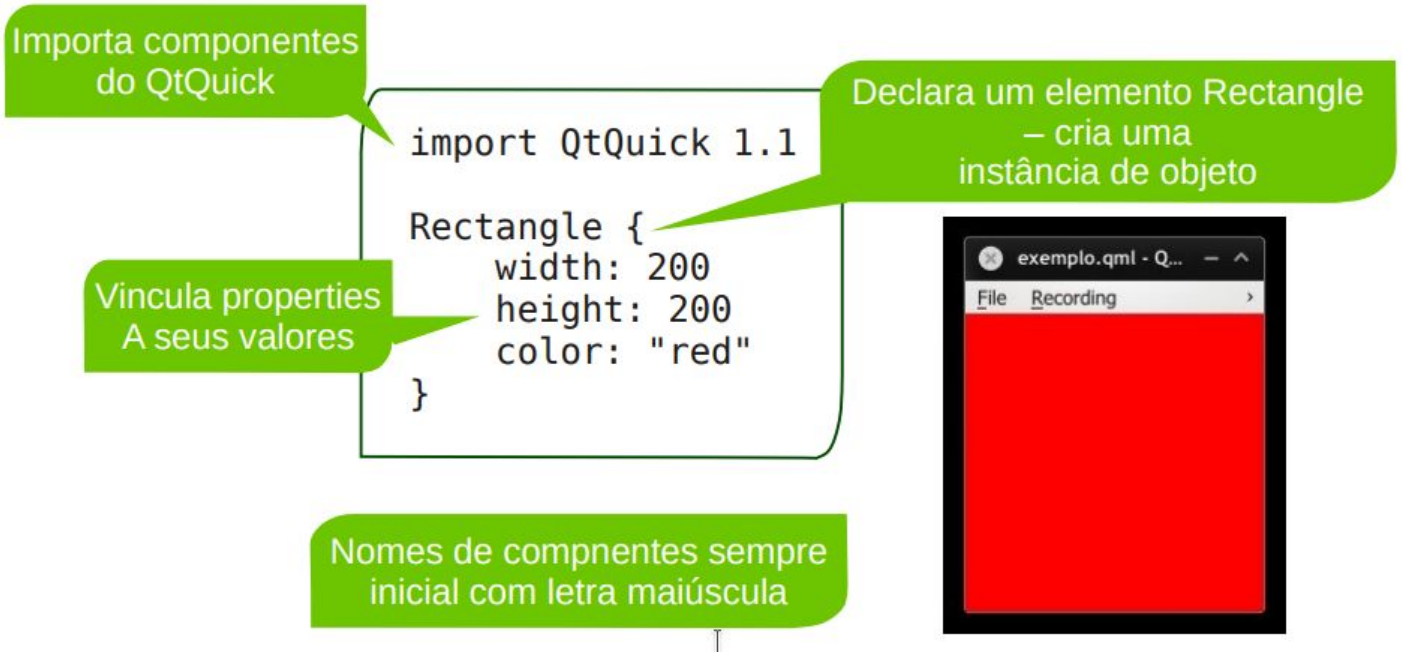
```
Item {
    propertyA: 100
    propertyB: propertyA + 100 // aqui

    onSomeEvent: myFunction() // aqui
}
```

O que é QML?

- **Qt Quick consiste de:**
  - QML – A linguagem
  - Qt Declarative – módulo Qt
  - Controls - conjunto de controles(Botões, Inputs, Textarea e etc.)
  - Contém o engine QML, context e view
  - Bindings Qt para QML – Mecanismos para integração entre C++ e QML
  - Ferramentas de depuração para *Qt Creator IDE*

O que é QML?

- **QML é uma linguagem declarativa baseada em Javascript**



Tipos disponíveis no QML

bool *Binary true/false value*

double - *Number with a decimal point, stored in double precision*

enumeration - *Named enumeration value*

int - *Whole number, e.g. 0, 10, or -20*

list - *List of QML objects*

real - *Number with a decimal point*

string - *Free form text string*

url - *Resource locator*

var - *Generic property type*

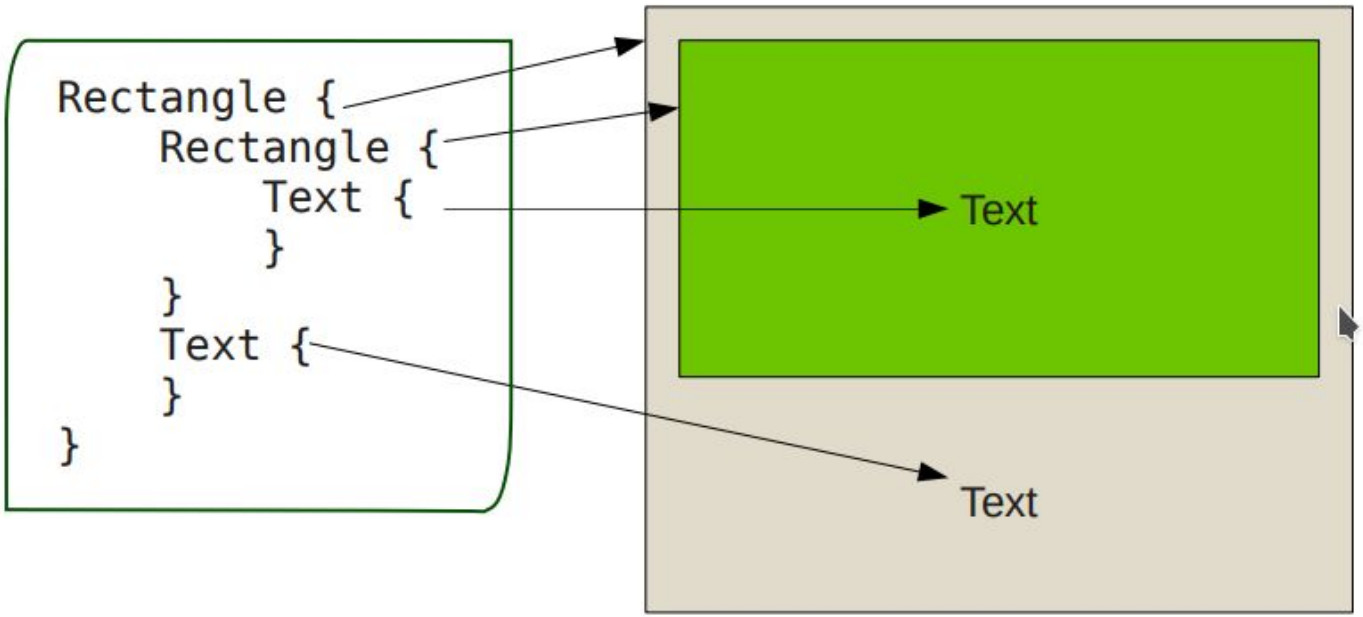
Importando Recursos

- **A diretiva *import* importa:**
  - Componentes de módulos C++
  - Outros módulos QML criados pelo usuário, libs e etc.
  - Arquivos Javascript
  - Quando importar módulos do Qt ou componentes C++, a versão tem que ser especificada

```
import QtQuick 1.1
import MyCppClasses 1.2
import "from-qml"
import "scripts.js"
```

Hierarquia de Objetos no QML

- **Declaração de elementos dentro de outros objetos**



Navegando na Árvore de Objetos

- **É possível se referir ao pai de um objeto usando a palavra chave *parent***

```
Rectangle {
    Rectangle {
        width: parent.width

        Text {
            color: parent.color
        }
    }
    Text {
    }
}
```

Nomeando Elementos

- **Pode-se nomear elementos utilizando a propriedade *id***

```
Rectangle {
    id: outerRectangle
    ...
```

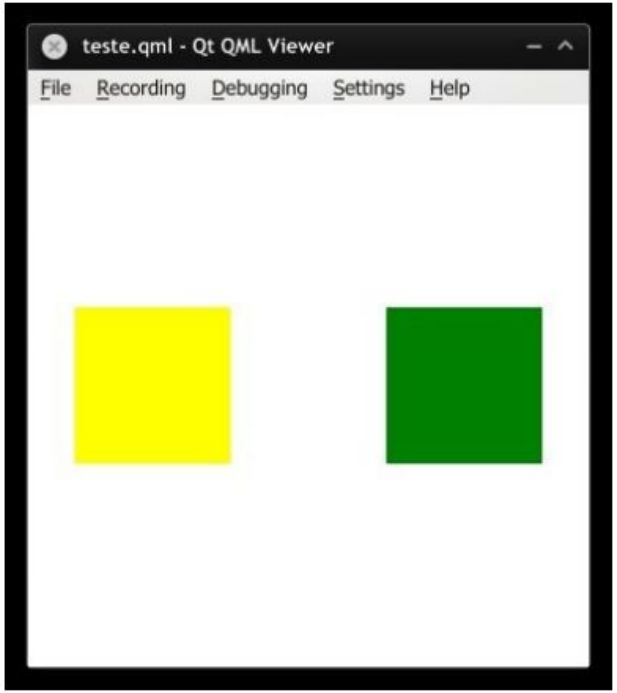
- **Você pode então referir a eles pelo *id***

```
{
    height: outerRectangle.height
    ...
```

Binding de Valores

- **Em QML, valores são amarrados e não atribuídos**

```
Rectangle {
    id: firstRect
    x: 10
    ...
}

Rectangle {
    x: 400 - firstRect.x
    ...
}
```



Qt provê um vasto conjunto de componentes

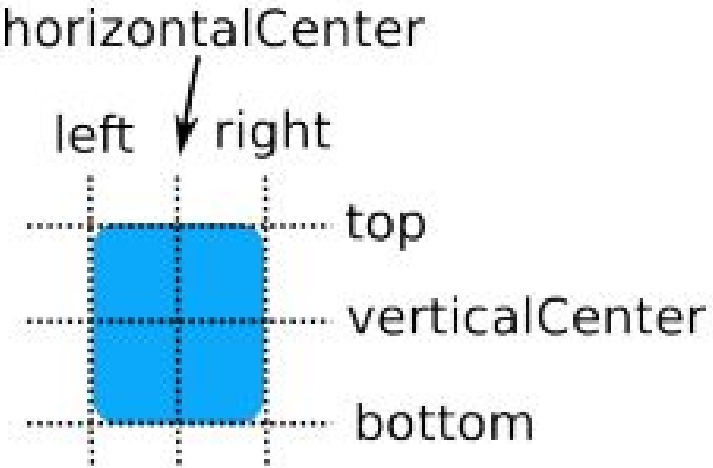
- **Visuais:**
  - *Rectangle*
  - *Text*
  - *Image*
  - *WebView*
  - *Button*
- **Não visuais:**
  - *Item*
  - *KeyboardDevice*
  - *ObjectModel*
- **Estruturais**
  - *Column*
  - *Row*
  - *GridLayout*

WebView - Renderizador de XHTML

```
import QtWebKit 1.1

WebView {
      url: "http://www.kde.org"
      preferredWidth: 1024
      preferredHeight: 768
      scale: 0.95
      smooth: false
}
```

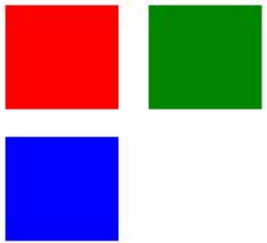
Propriedades de Layouts Baseados em Âncora

- **Pode-se ancorar ítens a:**
  - *Left, top, right, bottom*
  - *VerticalCenter, horizontalCenter*
  - *Baseline*
  - Pode-se especificar margens
  - individuais ou usar *anchors.margins*



Outros Layouts

- **Podemos criar layouts clássicos usando containers como *Grid*, *Row* e *Column*:**
  - Elementos serão posicionados conforme o *container* utilizado
  - A propriedade *spacing* estará disponível em todos os filhos



```
Grid {
    columns: 2
    spacing: 5

    Rectangle { width: 20; height: 20; color: "red" }
    Rectangle { width: 20; height: 20; color: "green" }
    Rectangle { width: 20; height: 20; color: "blue" }
}
```

Criando um Botão

- **É possível criar um botão usando *Rectangle, Text e MouseArea***

```
Rectangle {
    width: 200; height 100;
    color: "lightBlue"

    Text {
        anchors.fill: parent
        text: "Me pressione!"
    }

    MouseArea {
        anchors.fill: parent
        onClicked: { parent.color = "green" }
    }
}
```

Interação direta com Javascript

```
Rectangle {
    width: 200; height 100;
    color: "blue"

    Text {
        anchors.fill: parent
        text: "Me pressione!"
    }

    MouseArea {
        anchors.fill: parent
        onClicked: { parent.color = "green" }
    }
}
```

O quê aconteceu aqui?
Nós amaramos um função Javascript anônima a um signal.

Criando Componentes

- **Criar cada botão a partir de três elementos não é uma boa solução**

  - É possível criar componentes em QML usando qualquer elemento do Qt quick / Quick controls

  - Um componente pode ser instanciado como um elemento

  - Componentes podem ser mantidos em módulos que são incluídos em seus arquivos QML

Componente Botão

- **Crie o botão em um arquivo chamado *Button.qml***

```
import QtQuick 1.1

Rectangle {
    width: 200; height: 100;
    color: "lightBlue"
    property alias text: innerText.text

    Text {
        id: innerText
        anchors.fill: parent
    }

    MouseArea {
        anchors.fill: parent
        onClicked: { parent.color = "green" }
    }
}
```

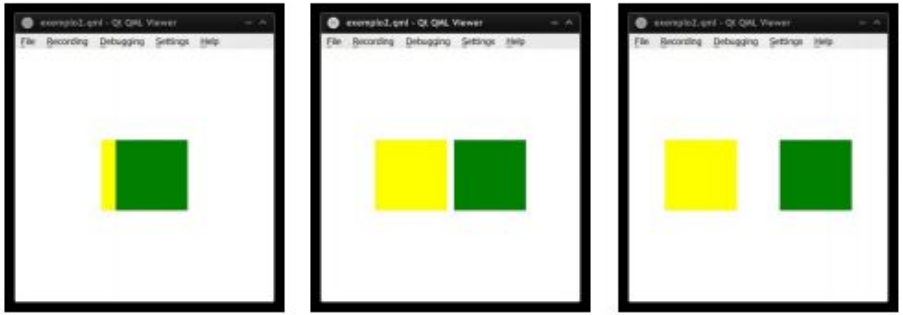
Componente Botão

- **Instancie os botões em seu arquivo QML**
  - O arquivo QML deve estar no mesmo diretório que o arquivo *Button.qml*

```
import QtQuick 1.1

Row {
    spacing: 10

    Button { text: "Oslo" }
    Button { text: "Copenhagen" }
    Button { text: "Helsinki" }
    Button { text: "Stockholm" }
}
```

Suporte a animação de elementos

```
Rectangle {
    id: firstRect
}

Rectangle {
    x: 400 – firstRect.x
    ...
}

SequentialAnimation {
    running: true
    loops: Animation.Infinite
    NumberAnimation { target: firstRect; property: "x"; to: 300 }
    NumberAnimation { target: firstRect; property: "x"; to: 50 }
}
```



“Amarrado”, mas não Atribuído

- Como QML amarra valores ao invés de atribuí-los, mudar uma propriedade de contexto a partir do C++ também muda seu valor no QML
- Serve também para propriedades entre elementos QML
- Evita *ifs* desnecessários através dos *binds*
- Evita propriedades globais através da associação a propriedades de outros elementos

Operações a partir de eventos

- Signals e Slots - Boa característica do QML
- É possível executar uma função quando um sinal for emitido
- Toda propriedade de um elemento qualquer QML emite um sinal quando ela mudar

```
Rectangle {
    property color previousColor
    property color nextColor
    onNextColorChanged: console.log("The next color will be: " + nextColor.toString())
}
```

Operações a partir de eventos

- Elemento ***Connections*** permite ligar sinais de um outro componente a um outro evento qualquer

```
MouseArea {
    id: area
}
// ...

Connections {
    target: area
    onClicked: foo(parameters)
}
```

QML para *Mobile*

- Um único código para diferentes sistemas
  - Android, iOS, Windows, Linux
- Ótimo desempenho e belo visual
- Componentes prontos - Quick Controls 2
  - oferece suporte ao Material Design do Google e *Universal Design* do Windows 10
- Qml Material do Papyros (GitHub)
  - inclui diversos componentes prontos para uso em Android e iOS
- Exemplo pronto (*Gallery*) no Qt for Android
  - ideal para começar um projeto

Mãos na massa!

- Vamos ao exemplo prático

Referências

- http://doc.qt.io/qt-5/qmltypes.html
- http://doc.qt.io/qt-5/qmlapplications.html
- http://doc.qt.io/qt-5/qmltypes.html