Qt Quick

Николай С Николов

30.10.2010

kubuntu

Qt, Qt Quick и QML

kubuntu

Qt е мултиплатформена библиотека за създаване на конзолни и графични приложения.

Qt Quick е колекция от технологии, които са разработени за да помогнат на разработчиците и дизайнерите да създават интуитивни, модерни и приятно изглеждащи интерфейси, каквито все повече се използват в мобилните телефони, плейъри и други мобилни устройства.

Qt Quick се състои от богат избор от графични елементи, декларативен език за описване на интерфейса и интерпретатор. Използва се колекция от C++ API-та, за да се интегрират тези елементи към стандартните Qt приложения.

Потребителският интерфейс и неговото държание се описва с QML, разширение на JavaScript, което позволява на разработчиците и дизайнерите да използват декларативен синтаксис. QML елементите са богат набор от графични и поведенчески блокове, които могат да се комбинират в QML документи, за да изграждат компоненти, вариращи от прости бутони до цели приложения с достъп до Интернет.

QML подобрява интеграцията между JavaScript и съществуващата в Qt QObject-базирана система от типове, добавя поддръжка за автоматично свързване на свойствата и предоставя мрежова прозрачност на ниво програмен език.

QML документ

kubuntu

Button.qml

```
import Qt 4.7

 Rectangle {
    property alias text: textItem.text

    width: 100; height: 30
    border.width: 1
    radius: 5
    smooth: true

    gradient: Gradient {
       GradientStop { position: 0.0; color:
"darkGray" }
       GradientStop { position: 0.5; color: "black" }
       GradientStop { position: 1.0; color:
"darkGray" }
    }

    Text {
       id: textItem
       anchors.centerIn: parent
       font.pointSize: 20
       color: "white"
    }

}
```

application.qml

```
import Qt 4.7

 Column {
    spacing: 10

    Button { text: "Apple" }
    Button { text: "Orange" }
    Button { text: "Pear" }
    Button { text: "Grape" }
 }
```

Property bindings

kubuntu

"Property binding"("Обвързване на свойствата") е декларативен начин за определяне на стойността на свойство на даден обект. Обвързването позволява стойността на свойство да се изрази чрез JavaScript израз, който дефинира стойност, свързана с други свойства или с данни, достъпни в приложението. Стойността на свойството автоматично се обновява, ако другите свойства или данни се променят.

QML разширява стандартният JavaScript енджин, така че всеки валиден JavaScript израз може да се използва за обвързване на свойства. Обвързванията могат да достъпват свойства на обекти, да извикват функции и дори да използват вградените в JavaScript обекти като Date и Math.

Ето някои примери за по-сложни обвързвания:

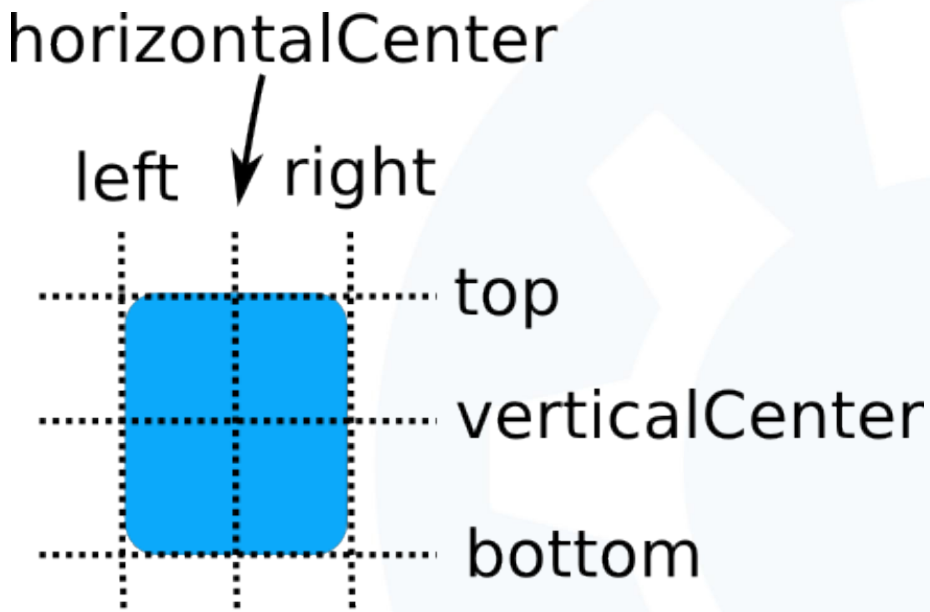
```
Rectangle {
    function calculateMyHeight() {
        return Math.max(otherItem.height, thirdItem.height);
    }

    anchors.centerIn: parent
    width: Math.min(otherItem.width, 10)
    height: calculateMyHeight()
    color: { if (width > 10) "blue"; else "red" }
}
```

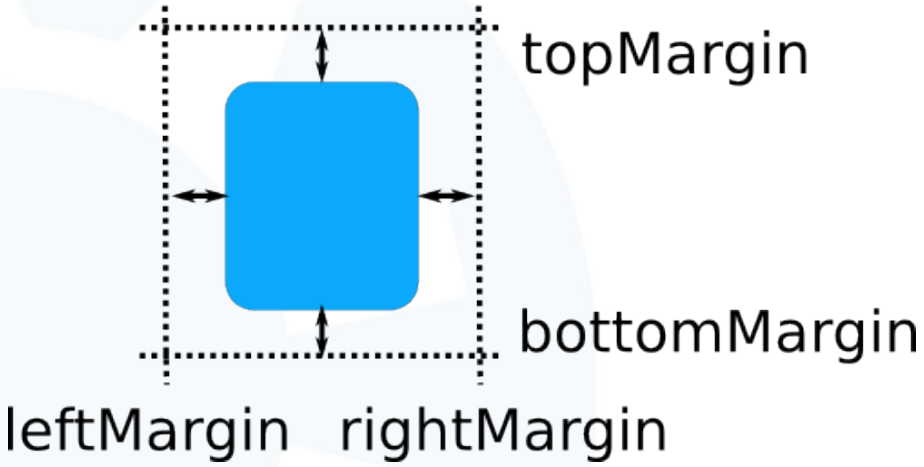
Присвояването на стройност през JavaScript не създава property binding.

Подреждане на елементите

kubuntu

- Чрез обвързване
- Чрез котви






```
Rectangle {
    id: rect1; ...
}
Rectangle {
    id: rect2;
    anchors.left: rect1.right; ...
    anchors.leftMargin: 5; ...
}
```

Състояния (States)

kubuntu

Промени, които може да се направят между състояния:

- Да се покажат някои елементи и да се скрият други
- Представете различни налични действия за потребителя
- Пускане, спиране или пауза на анимации
- Изпълнение на скрипт, изискван от новото състояние
- Промяна на свойство на даден елемент
- Показване на друг изглед на екрана

Промените между състоянията могат да се анимират чрез "преходи"

```
import Qt 4.7

Rectangle {
    id: myRect
    width: 200; height: 200
    color: "red"

    MouseArea {
        anchors.fill: parent
        onClicked: myRect.state = 'moved'
    }

    states: [
        State {
            name: "moved"
            PropertyChanges { target: myRect; x: 50; y:
50 }
        }
    ]
}
```

Преходи (Transitions)

Преходите се използват за да се опишат анимации, които да бъдат приложени, когато има промяна на състоянията.

```
import Qt 4.7

 Rectangle {
    id: rect
    width: 100; height: 100
    color: "red"

    MouseArea {
       anchors.fill: parent
       onClicked: rect.state = "moved"
    }

    states: State {
          name: "moved"
       PropertyChanges { target: rect; x: 50; y: 50 }
    }

     transitions: Transition {
       PropertyAnimation { properties: "x,y"; duration: 1000 }
     }
 }
```

Анимации

В QML, анимациите се използват за да се опише плавна промяна на дадено свойство. Те могат да се прилагат по различни начини в зависимост от контекста, в който се използват.

```
import Qt 4.7

 Rectangle {
    width: 100; height: 100
    color: "red"

    PropertyAnimation on x { to: 50; duration: 1000; loops: Animation.Infinite }
    PropertyAnimation on y { to: 50; duration: 1000; loops: Animation.Infinite }
 }
```

Behaviors

kubuntu

Често анимация следва да се прилага, когато дадено свойство промени стойността си. В този случай, Behavior може да се използва, за да зададете по подразбиране анимация при промяната му.

```
import Qt 4.7

Item {
    width: 100; height: 100

    Rectangle {
        id: rect
        width: 100; height: 100
        color: "red"

        Behavior on x { PropertyAnimation { duration: 500 } }
        Behavior on y { PropertyAnimation { duration: 500 } }
    }

    MouseArea {
        anchors.fill: parent
        onClicked: { rect.x = mouse.x; rect.y = mouse.y }
    }
}
```

Бъдеще

- QML 3D
- Graph scene

Въпроси