QtQuick Training Course

Module Two

Module Two

Objectives

1 Positioning elements

Absolute positioning

Relative positioning

Anchors

2 Making things move

How to create States

Set Transitions and Animations

All kinds of easings and animations

Objectives

3 QtQuick and Javascript are good friends

Declarative and imperative together

Creating javascript functions in a QtQuick file

Importing a javascript file

Component Oriented Programming in QtQuick

Topics

1 Positioning elements

2 Making things move

3 QtQuick and Javascript are good friends

4 Questions

5 Lab

Positioning elements

Absolute positioning

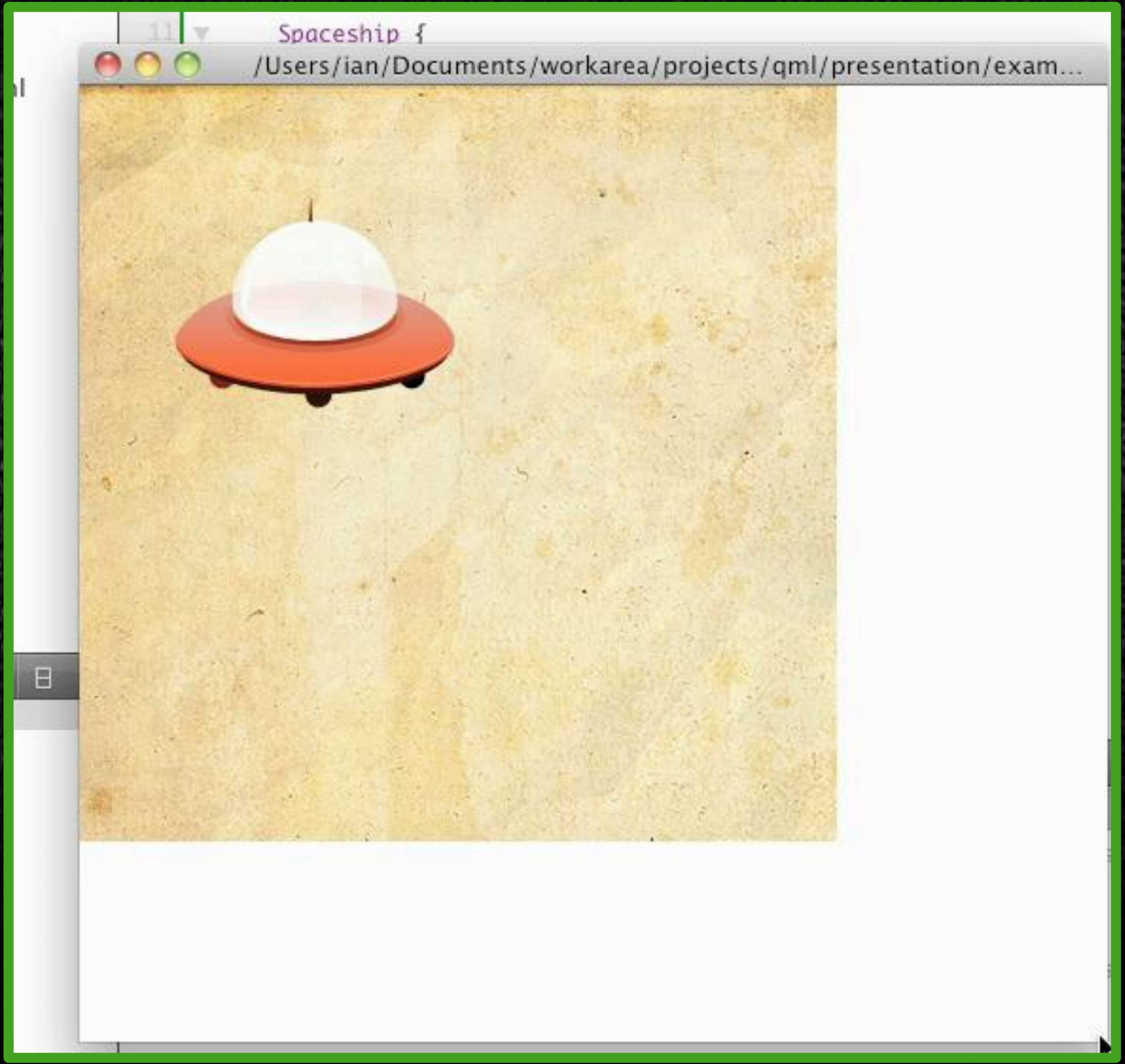
Define the item's position and size relative to its parent

x, y, width and height

```
Item {
   width: 400
   height: 400

   Image {
      source: "images/background.png"


   Spaceship {
      id: spaceship
      x: 50
      y: 60
   }
}
```


Spaceship {
/Users/ian/Documents/workarea/projects/qml/presentation/exam...

See video: addon/module-002/videos/basic-positioners.mov

See example: addon/module-002/examples/basic-positioners.qml

Positioning elements

Relative positioning

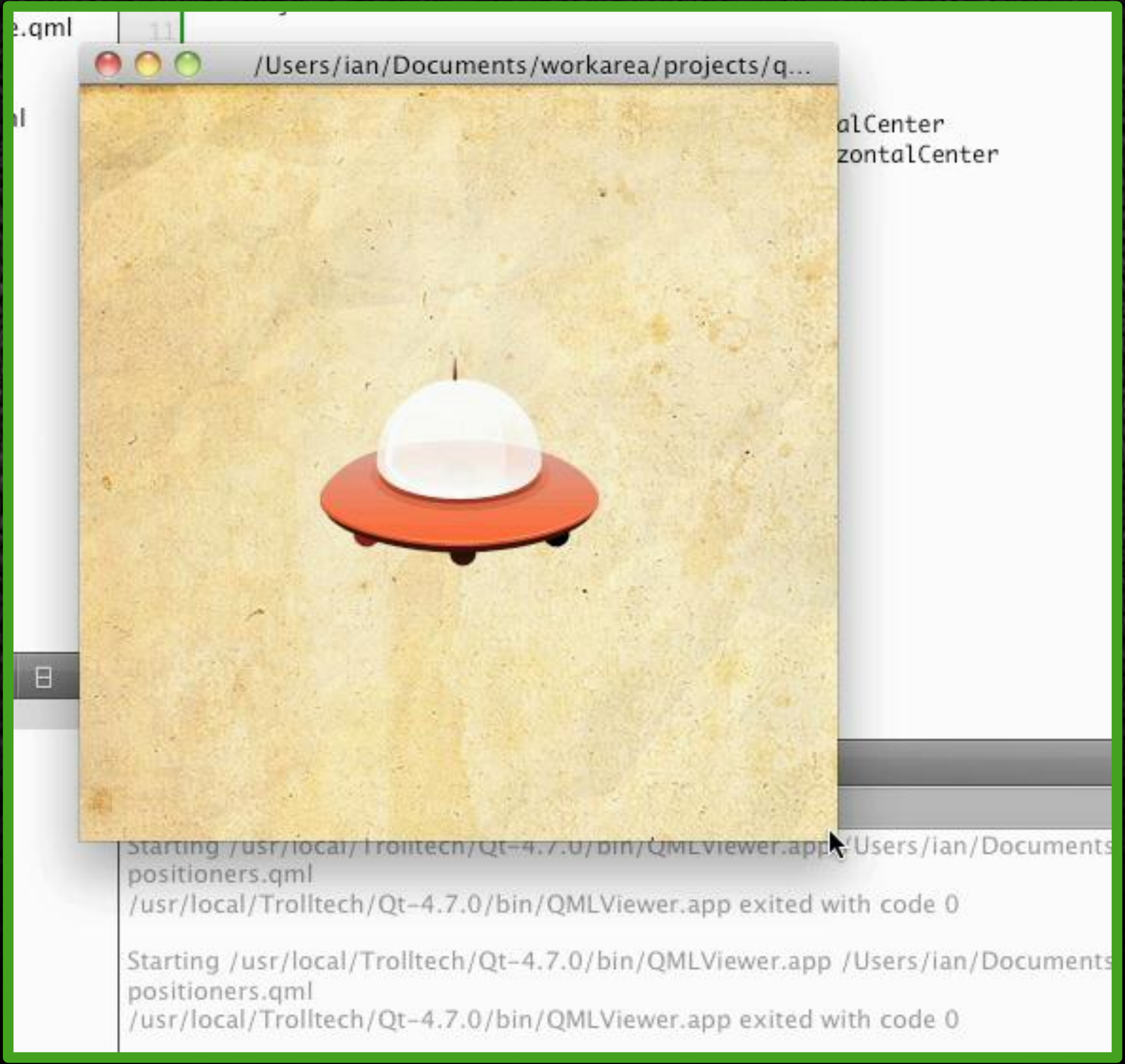
Anchors provide a way to position an item by specifying its relationship with other items

anchors

```
Item {
    width: 400; height: 400

    Image {
        anchors.fill: parent
        source: "images/background.png“
    }

    Spaceship {
        id: spaceship
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter
    }
}
```



```
/Users/ian/Documents/workarea/projects/q...
```

```
alCenter
zontalCenter
```

```
Starting /usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app /Users/ian/Documents
positioners.qml
/usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app exited with code 0

Starting /usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app /Users/ian/Documents
positioners.qml
/usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app exited with code 0
```

See video: addon/module-002/videos/anchors-positioners.mov

See example: addon/module-002/examples/anchors-positioners.qml

More about anchors

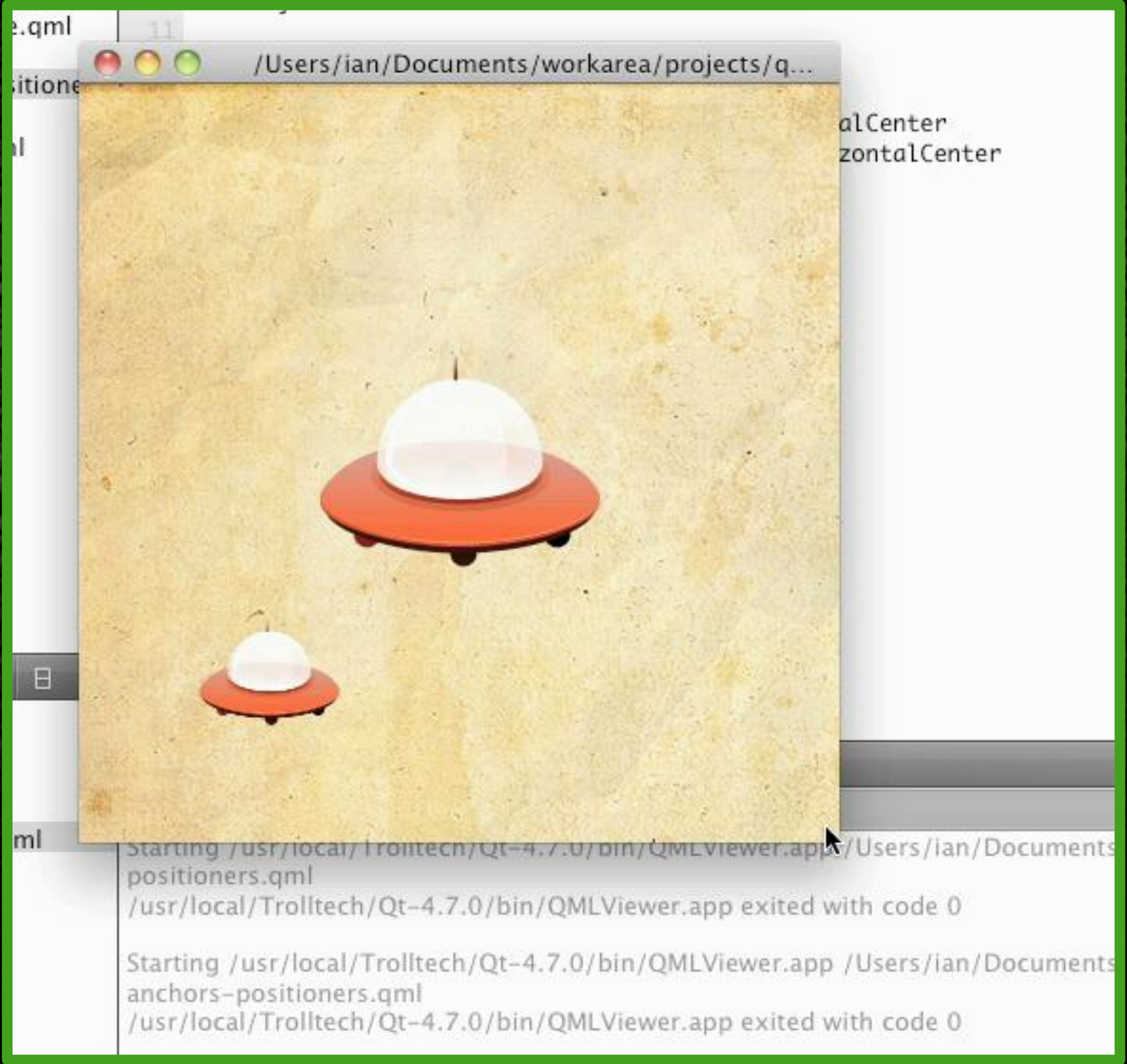
There are many ways to specify how an item is related to another

anchors.right, anchors.rightMargin ...

```
Item {
    width: 400
    height: 400

    Spaceship {
        id: spaceship
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter

    Spaceship {
        anchors.top: spaceship.bottom
        anchors.right: spaceship.right
        anchors.rightMargin: 100
    }
}
```



```
Starting /usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app /Users/ian/Documents
positioners.qml
/usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app exited with code 0

Starting /usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app /Users/ian/Documents
anchors-positioners.qml
/usr/local/Trolltech/Qt-4.7.0/bin/QMLViewer.app exited with code 0
```

See video: addon/module-002/videos/more-anchors.mov

See example: addon/module-002/examples/more-anchors-positioners.qml

Topics

How to create States

The State element defines configurations of objects and properties.

```
Item {
    id: myItem
    width: 400
    height: 400

    Image {
        id: spaceship
        source: "images/spaceship.png"
        x: 10
        y: 50
    }

    states: [
        State {
            name: "leftXMove"
            PropertyChanges {
                target: spaceship
                x: 200
            }
        }
    ]
}
```

How to create States

You can create as many states as you need for an object

```
...
states: [
    State {
        name: "leftXMove"
        PropertyChanges {
            target: "spaceship"
            x: 200
        }
    },
    State {
        name: "downYMove"
        PropertyChanges {
            target: "spaceship"
            y: 90
        }
     }
  ]
}
...
```

All properties not expressed will be the same as the base state

The purpose of creating States

It is easy to change from one state to another

```
Image {
   id: button; source  "images/button.png"
   y: 50  x: 10
   MouseArea {
      anchors.fill: parent; onClicked: myItem.state = 'leftXMove'
   }
}
```

Executing a function to set a different state string name

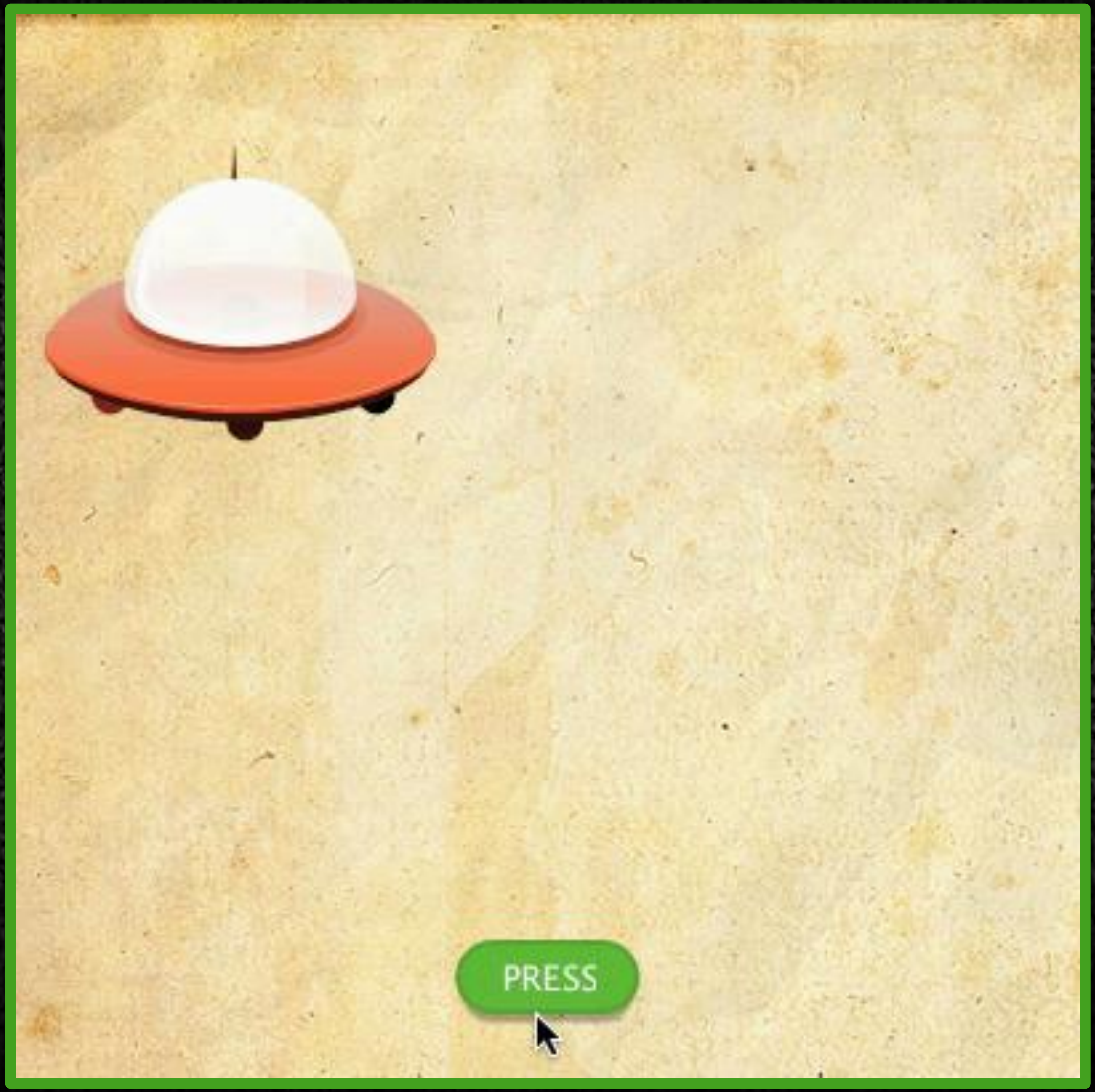
or

```
Image {
   id: button; source  "images/button.png"
   y: 50; x: 10
   MouseArea {
      id:mouseArea; anchors.fill: parent
   }
}
states: State {
   name: "leftXMove"; when: mouseArea.clicked
   PropertyChanges { target: myItem; x: 200 }
 }
```

Using “when” method. It will change the property but inside the state element

The purpose of creating States

This is the result ...



See video: addon/module-002/videos/spaceship-no-motion.mov

The purpose of creating States

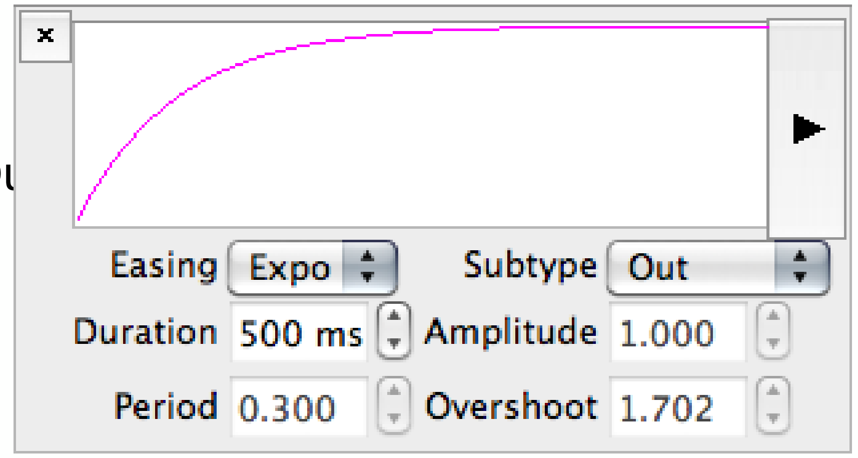
... but this one is much more interesting

See video: addon/module-002/videos/spaceship-motion.mov

Animating from one State to another

You just need to add a transition element to animate between states

```
...
transitions: [
    Transition {
        from: ""
        to: "leftXMove"
        NumberAnimation {
            properties: "x, y"
            duration: 500;
            easing.type: Easing.Ou
        }
    }
]
...
```



Easing Expo
Subtype Out
Duration 500 ms
Amplitude 1.000
Period 0.300
Overshoot 1.702

Helper: Qt Quick Toolbar

The first state is an empty string by default

See example: addon/module-002/examples/animation-example.qml

Main transition and animation elements

from and to

the element’s initial and final state string

target

the animated element’s id

properties

the property that you want to change during the animation. This can be an array of properties

easing.type

choose an easing curve to add a specific effect to your animation

To know more about different animation elements:
http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeelements.html

Animation types

There are many ways to achieve your needs

NumberAnimation

ParallelAnimation

SequentialAnimation

PauseAnimation

RotationAnimation

To know more about different animation types:
http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeelements.html

NumberAnimation

This allows you to animate changes in a real number type property

```
...
NumberAnimation {
    properties: "x, y"
    duration: 500;
    easing.type: Easing.OutExpo;
}
...
```

This is the basic animation element. Most animations are about changing numbers.

Parallel and Sequential

You can animate specific properties in a specific order

ParallelAnimation

```
ParallelAnimation {
   NumberAnimation {
      target :  myRect
      properties: "x"
      duration: 500;
      easing.type: Easing.OutExpo;
   }

   NumberAnimation {
      target :  myRect
      properties: "y"
      duration: 500;
      easing.type: Easing.OutExpo;
   }
}
```

SequentialAnimation

```
SequentialAnimation {
   NumberAnimation {
      target :  myRect
      properties: "x"
      duration: 500;
      easing.type: Easing.OutExpo;
   }

   NumberAnimation {
      target :  myRect
      properties: "y"
      duration: 500;
      easing.type: Easing.OutExpo;
   }
}
```

Parallel and Sequential

You can animate specific properties in a specific order

ParallelAnimation

See video: addon/module-002/videos/parallel.mov

SequentialAnimation

See video: addon/module-002/videos/sequential.mov

Other animation elements

The RotationAnimation allows you to add specific rotation properties to your animation

```
states: {
    State {
        name: "180";
        PropertyChanges { target: myItem; rotation: 180 }
    }
    State {
        name: "-90"
        PropertyChanges { target: myItem; rotation: -90 }
    }
}
transition: Transition {
    RotationAnimation {
        direction: RotationAnimation.Shortest
    }
}
```

Other animation elements

The PauseAnimation allows you to add delays in your animation

```
PauseAnimation {
    target :  myRect
    duration: 100
}
```

Animating with behavior

```
Rectangle {
    width: 20; height: 20; color: "#00ff00"
    y: 200
    Behavior on y {
        NumberAnimation {
            easing.type: Easing.OutBounce
            duration: 200
        }
    }
}
```

Topics

Declarative and Imperative together

Qt Quick lists elements with properties, and JavaScript allows you to express more complex behavior than static values

```
Item {
    id: label1
    x: 80
    width: 100
    height: 100

    Image {
        source: {
            if(pressed) {
                return "img2.png";
            }  else {
                return "img1.png";
            }
        }

}
```

Javascript function inside QtQuick

You can add a javascript function anywhere in your QtQuick file

```
function randomState()
{
    var statesArray = ["topLeft"  "topRight"  "bottomLeft"  "bottomRight"];
    var randomNumber = Math.floor(Math.random()*statesArray.length);
    return statesArray[randomNumber];
}
```

This is a simple function that picks a number and references it in an array of states

See example: addon/module-002/examples/javascript-example.qml

Javascript function inside QtQuick

This is the result after adding this function to the previous example

See video: addon/module-002/videos/javascript.mov

See file reference: addon/module-002/examples/javascript-example.qml

Importing a javascript file

If you want to organize your code, you can import a js file to QtQuick

```
import QtQuick 1.0
import "random.js" as RandomFunction

Item {
    id: myItem
    width: 400

...

    MouseArea {
        anchors.fill: parent
        onClicked: {
            myItem.state = RandomFunction.randomState();
        }
    }
}
```

See example: addon/module-002/examples/importing-javascript.qml

Component Programming in QtQuick

It’s a good idea to recycle your code and create components for elements that can be reused. An example? A button!

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

}
```

See example: addon/module-002/examples/reusing-button.qml

Component Programming in QtQuick

Now there is a reusable button layout, but some improvements are needed. The onClicked mouse event needs to be inside Button.qml

reusing-button.qml

```
...
Button {
    id: button
    labelText: "PRESS"
    anchors.horizontalCenter: myItem.horizontalCenter
    anchors.bottom: myItem.bottom
    anchors.bottomMargin: 20

    MouseArea {
        anchors.fill: parent
        onClicked: {myItem.state = RandomFunction.randomState();}
    }
}
...
```

See example: addon/module-002/examples/reusing-button.qml

Component Programming in QtQuick

There are default properties in QtQuick that allows you to have communication between classes

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText
    signal buttonClicked

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

}
```

These properties are declared in the beginning of the file

Component Programming in QtQuick

When the button is clicked a signal is emitted. Now, all you need to do is interpret it

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText
    signal buttonClicked

    MouseArea {
        anchors.fill: parent
        onClicked: {
            button.buttonClicked();
        }
    }
}
```

Component Programming in QtQuick

The MouseArea control is now inside the Button class and a function is executed when the signal is emitted

reusing-button.qml

```
...
Button {
    id: button
    labelText: "PRESS"
    anchors.horizontalCenter: myItem.horizontalCenter
    anchors.bottom: myItem.bottom
    anchors.bottomMargin: 20

    onButtonClicked: {
        myItem.state = RandomFunction.randomState();
    }
}
...
```

See example: addon/module-002/examples/reusing-button.qml

Module Two

Topics

Questions

How do you create animations between states?

What is the difference between Sequential and Parallel

Animations?

What happens if I don’t declare the from and to transition properties?

How do I create a javascript function inside a QtQuick file?

How do I execute a function in QtQuick from an imported js file?

Is it possible to reuse code in QtQuick?

What is a signal?

Topics

1 Positioning elements

2 Making things move

3 QtQuick and Javascript are good friends

4 Questions

5 Lab

Lab

Spaceship attack! Reproduce the movement below

See video: addon/module-002/videos/spaceship-attack.mov

Optional: The spaceships must be a component file

See lab: addon/module-002/labs/lab-animation/labTwo.qmlproject

(c) 2011 Nokia Corporation and its Subsidiary(-ies).