QtQuick Training Course

Module One

Module One

Objectives

1 Things to know about Qt

What is it?

Why use it?

Who uses it?

2 Getting started with QtQuick

Declarative UI

Syntax

Examples

Qt Creator IDE

Comparison between languages

Hello world

Objectives

3 Layout and Interaction

- Item
- Rectangle
- Image
- Text
- Mouse Events
- Key Events

Module One

Topics

1 Things to know about Qt

2 Getting started with QtQuick

3 Layout and Interaction

4 Questions

5 Lab

What is Qt?

Cross-platform framework

For GUI and non-GUI apps

Used everywhere (Desktop, web and embedded development)

LGPL licensed in 2009

Free for anyone to use it: http://qt.nokia.com/

Why Qt?

Qt is intuitive

Just one code to all platforms

Three licenses to fit your needs (Commercial, LGLP or GPL)

Huge community support

Provides free tools to start learning

Used by 250.000 developers (commercial and open source)

Qt Applications

Autodesk software (Maya and 3d Studio)

Google Earth

Skype for Linux

Opera

Full support for Nokia devices

Module One

Topics

1 Things to know about Qt

2 Getting started with QtQuick

3 Layout and Interaction

4 Questions

5 Lab

Qt Quick is

QML (language)

QtDeclarative (native module, C++ API)

Qt Creator

```
Rectangle {
    width: 200
    height: 200
    Text {
        x: 66
        y: 93
        text: "Hello World"
    }
}
```

See example: addon/module-001/examples/hello-world.qml

Tools

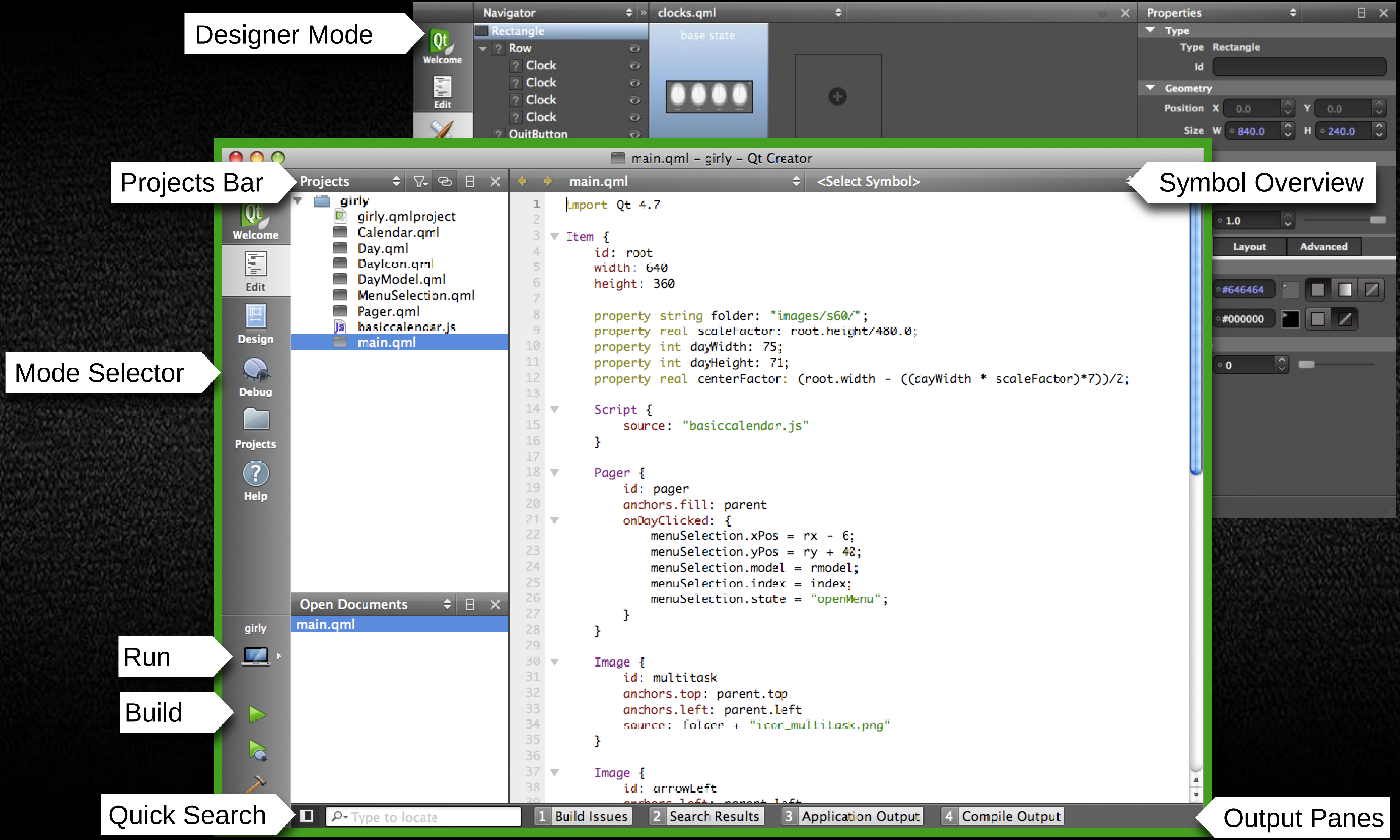
Qt 4.7

http://qt.nokia.com/downloads

Qt Creator IDE

- Visual editor for creating QML components (Qt Quick Designer)
- Project and build management tools
- Integrated, context-sensitive help system
- Visual debugger
- Rapid code navigation tools

Qt Creator



Designer Mode

Projects Bar

Mode Selector

Run

Build

Quick Search

Symbol Overview

Output Panes

```
import Qt 4.7

Item {
    id: root
    width: 640
    height: 360

    property string folder: "images/s60/";
    property real scaleFactor: root.height/480.0;
    property int dayWidth: 75;
    property int dayHeight: 71;
    property real centerFactor: (root.width - ((dayWidth * scaleFactor)*7))/2;

    Script {
        source: "basiccalendar.js"
    }

    Pager {
        id: pager
        anchors.fill: parent
        onDayClicked: {
            menuSelection.xPos = rx - 6;
            menuSelection.yPos = ry + 40;
            menuSelection.model = rmodel;
            menuSelection.index = index;
            menuSelection.state = "openMenu";
        }
    }

    Image {
        id: multitask
        anchors.top: parent.top
        anchors.left: parent.left
        source: folder + "icon_multitask.png"
    }

    Image {
        id: arrowLeft
```

Comparison between languages

Actionscript: MenuButton.as

```
public class MenuButton extends MovieClip
      public function MenuButton() {
              this.x = 60;
              this.addEventListener(MouseEvent.MOUSE_DOWN,
ClickBt);
       }
       function ClickBt(e:MouseEvent) {
              trace("clicked");
       }
}
```

QtQuick: MenuButton.qml

```
Item {
   x:60;
   MouseArea: {
         anchors.fill: parent;
         onClicked: print("clicked");
   }
}
```

Module One

Topics

1 Things to know about Qt

2 Getting started with QtQuick

3 Layout and Interaction

4 Questions

5 Lab

Layout

Item

Rectangle

Image

Text and TextInput

Item

The base of all visual elements in QtQuick

Item has no visual appearance

It defines all the properties that are common across visual items

Common properties examples: anchors, clip, width, height, opacity, x, y, rotation or scale

Item code snippet

The basic of all visual elements in QtQuick

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

Anchors

```
Item {
    id: label2
    anchors.left: label1.left
    anchors.top: label1.top
    anchors.topMargin: 30
}
```

Use the Anchor property instead of hardcoding the x and y. On module two, there will be more explanation about anchors

Layout

Item

Rectangle

Image

Text and TextInput

Rectangle

A Rectangle is painted using a solid color and an optional border.

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

    Rectangle {
       anchors.fill: parent
       color: "red"
       border.color: "black"
       border.width: 5
       radius: 10
    }
}
```

You can use the radius property to create rounded borders.

See example: addon/module-001/examples/item-example.qml

Layout

Item

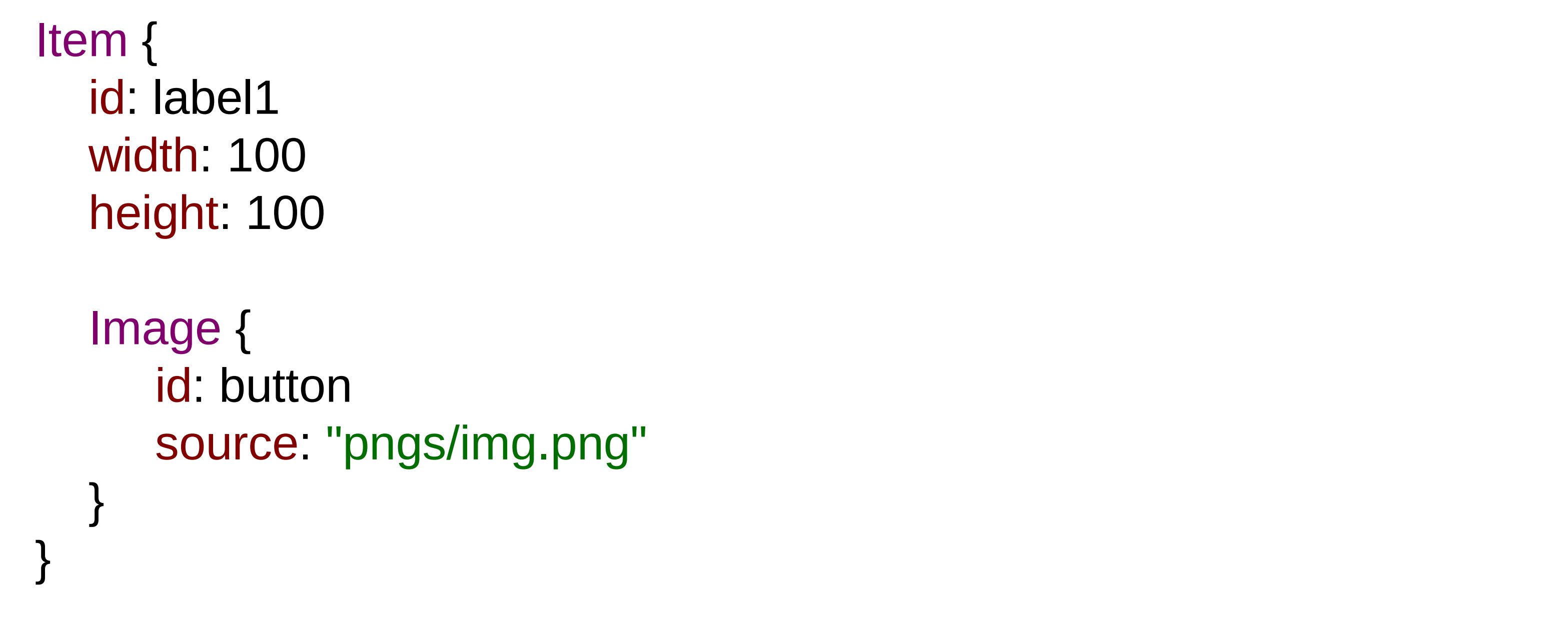
Rectangle

Image

Text and TextInput

Image

This element allows you to add bitmap to a scene.



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

    Image {
        id: button
        source: "pngs/img.png"
    }
}
```

It is a good practice not to hardcode the image width and height. QtQuick will automatically do this job.

To know more about which formats are supported:
http://doc.qt.nokia.com/4.7-snapshot/qml-image.html

See example: addon/module-001/examples/image-example.qml

Layout

Item

Rectangle

Image

Text and TextInput

Text and TextInput

This item allows you to add formatted texts to a scene.

```
Text {
    id: text1
    text: "Hello World!"
    font.family: "Helvetica"
    font.pixelSize: 18
    color: "red"
}
```

```
TextInput {
    id: input
    color: "red"
    text: "Default Text"
    width: 200; height: 24
    focus: true
}
```

To know more about different text properties:
http://doc.qt.nokia.com/4.7-snapshot/qml-text.html

Interaction

- MouseArea
- FocusScope
- Flickable

MouseArea

This item handles mouse events for items that are on the scene.

```
Rectangle {
    width: 100; height: 100
    color: "green"

    MouseArea {
        anchors.fill: parent
        onClicked: { parent.color = 'red' }
    }
}
```

MouseArea

Drag an item in a determined space to create a scrollbar.

```
Rectangle {
    id: slider;
    width: 320; height: 40;
    color: "green"
    Rectangle {
        id: handle; width: 40; height: 40
        color: "red"
        MouseArea {
            anchors.fill: parent
            drag.target: parent; drag.axis: "XAxis"
            drag.minimumX: 0
            drag.maximumX: slider.width - handle.width
        }
    }
}
```

See example: addon/module-001/examples/scrollbar-mousearea.qml

MouseArea

All Signals

onCanceled

onClicked

onDoubleClicked

onEntered

onExited

onPositionChanged

onPressAndHold

onPressed

onReleased

To know more about MouseArea signals:
http://doc.qt.nokia.com/4.7-snapshot/qml-mousearea.html

Interaction

MouseArea

FocusScope

Flickable

Key focus

You can generate a key event when a key is pressed.

```
Rectangle {
    color: "lightsteelblue"; width: 240; height: 25
    Text { id: myText }
    Item {
        id: keyHandler
        focus: true
        Keys.onPressed: {
            if (event.key == Qt.Key_A)
                myText.text = 'Key A was pressed'
            else if (event.key == Qt.Key_B)
                myText.text = 'Key B was pressed'
            else if (event.key == Qt.Key_C)
                myText.text = 'Key C was pressed'
        }
    }
}
```

See example: addon/module-001/examples/focusscope-event.qml

Key navigation

It is common in key-based UIs to use arrow keys to navigate between focused items

```
Grid {
    columns: 2
    width: 100; height: 100
    Rectangle {
       id: item1
       focus: true
       width: 50; height: 50
       color: focus ? "red" : "lightgray"
       KeyNavigation.right: item2
    }
    Rectangle {
       id: item2
       width: 50; height
       color: focus ? "red" : "lightgray"
       KeyNavigation.left: item1
    }
}
```

To know more about keyboard focus:
http://doc.qt.nokia.com/4.7-snapshot/qdeclarativefocus.html

See example: addon/module-001/examples/key-navigation.qml

Interaction

MouseArea

FocusScope

Flickable

Flickable

All elements placed in a Flickable item can be dragged and flicked.

```
Flickable {
    width: 200; height: 200
    contentWidth: image.width
    contentHeight: image.height

    Image { id: image; source: "bigImage.png" }
}
```

For fine-tuning your needs:
http://doc.qt.nokia.com/4.7-snapshot/qml-flickable.html

See example: addon/module-001/examples/flickable-area.qml

Topics

1. Things to know about Qt
2. Getting started with QtQuick
3. Layout and Interaction
4. Questions
5. Lab

Questions

What is Qt and QtQuick?

How is a QtQuick component file structured?

What is the basic visual element in QtQuick?

How can you position an item relatively to another?

How can you handle mouse events in QtQuick?

Name some QtQuick elements that inherits from Item.

How many components can you declare in a QtQuick file?

Module One

Topics

1 Things to know about Qt

2 Getting started with QtQuick

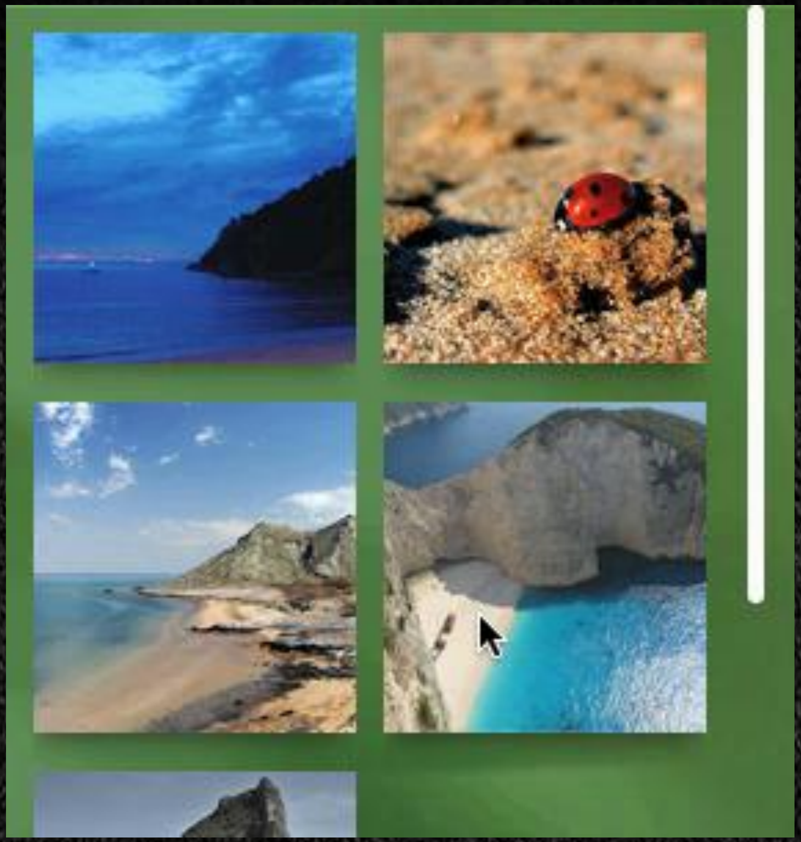
3 Layout and Interaction

4 Questions

5 Lab

Lab

Align items inside a Flickable area. Use anchors to do the job



See video: addon/module-001/videos/lab-video-example.mov

Optional: Create a scrollbar for it

See lab: addon/module-001/labs/lab-flickable/LabOne.qmlproject