Qt Quick for Qt Developers

Introduction to Qt Quick

Based on Qt 5.4 (QtQuick 2.4)

Contents

The Qt Company

- 30,000 feet Qt overview
- Meet Qt Quick
- Concepts

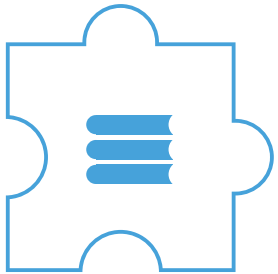
Objectives

The Qt Company

- Overview of the Qt library
  - Qt framework presentation
  - Qt Quick inside the Qt framework
- Understanding of QML syntax and concepts
  - Elements and identities
  - Properties and property binding
- Basic user interface composition skills
  - Familiarity with common elements
  - Understanding of anchors and their uses
  - Ability to reproduce a design

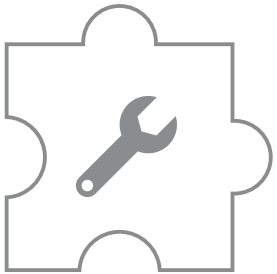
The Leading C++ Cross-Platform Framework

The Qt Company



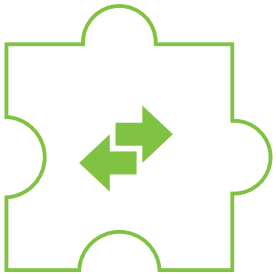
Cross-Platform Class Library

One Technology for All Platforms



Integrated Development Tools

Shorter Time-to-Market



Cross-Platform IDE, Qt Creator

Productive development environment

Used by over 800,000 developers in 70+ industries

Proved & tested technology – since 1994

Qt UI Offering – Choose the Best of All Worlds

The Qt Company

Qt Quick

C++ on the back, declarative UI design (QML) in the front for beautiful, modern touch-based User Experiences.

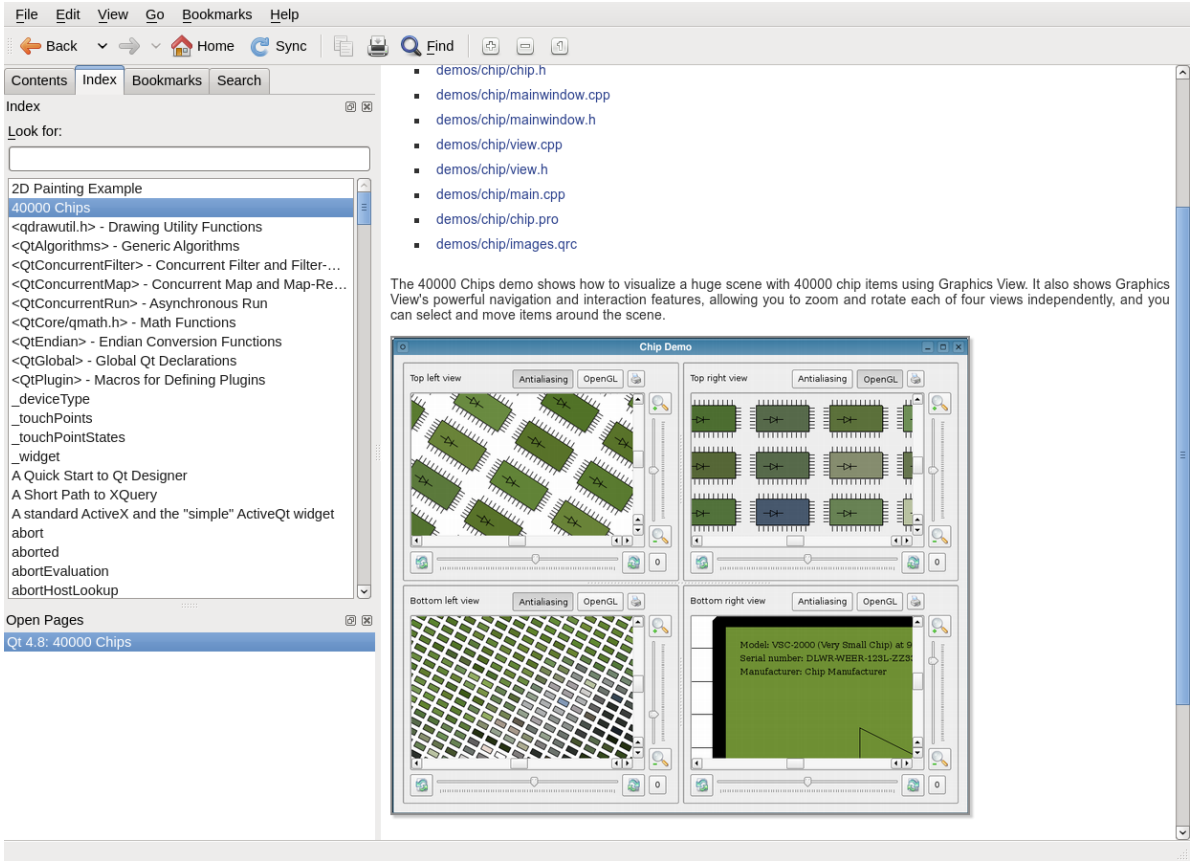
Qt Widgets

Customizable C++ UI controls for traditional desktop look-and-feel. Also good for more static embedded UIs for more limited devices / operating systems.

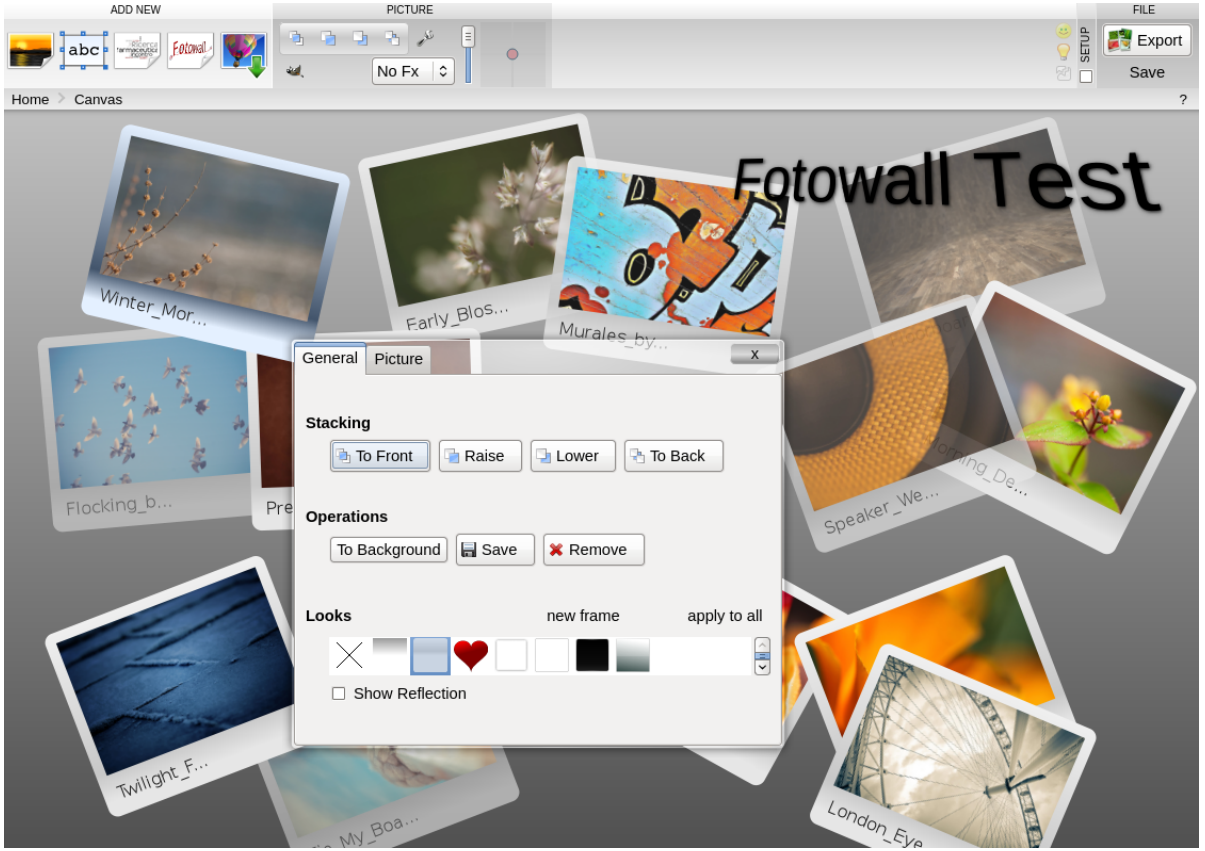
Web / Hybrid

Use HTML5 for dynamic web documents, Qt Quick for native interaction.

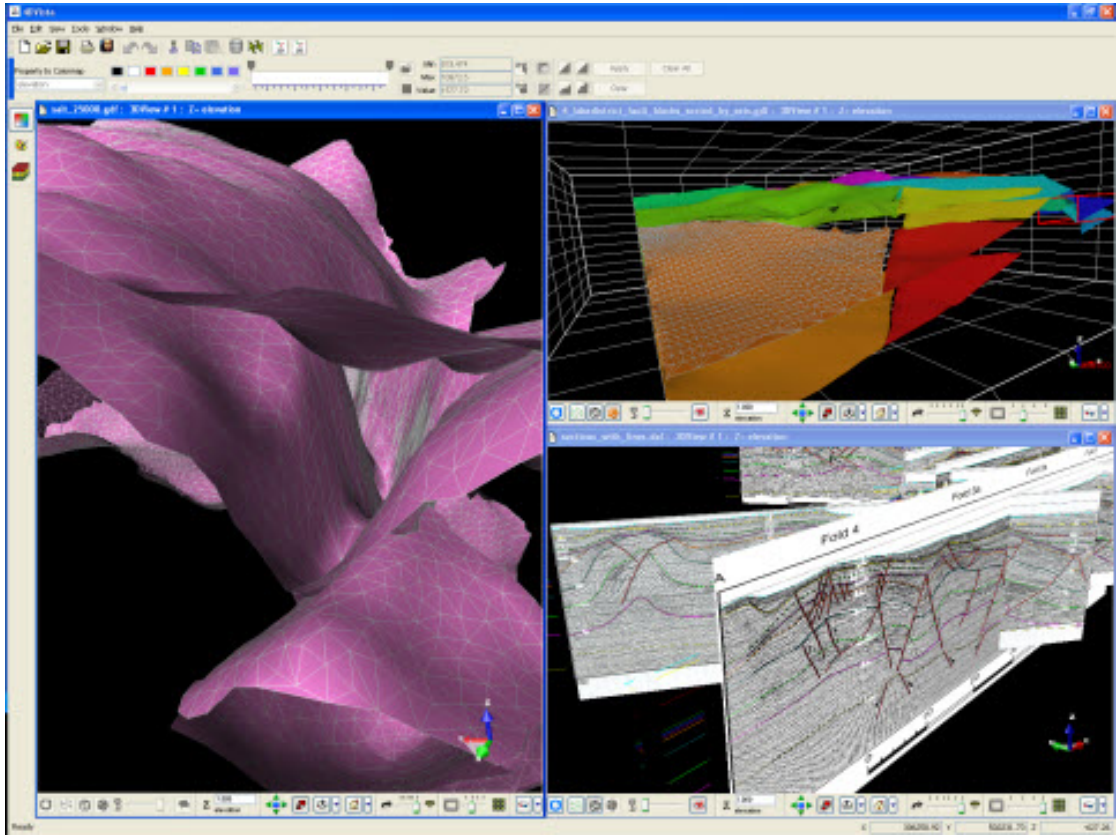
The Widget World



The Graphics View World

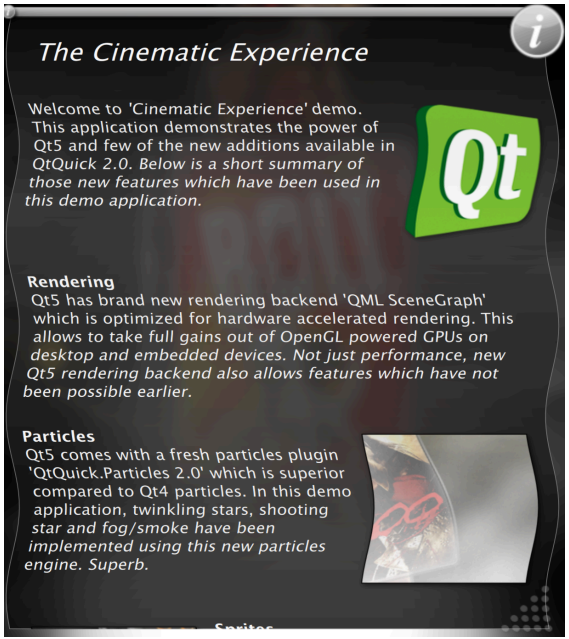


The OpenGL World



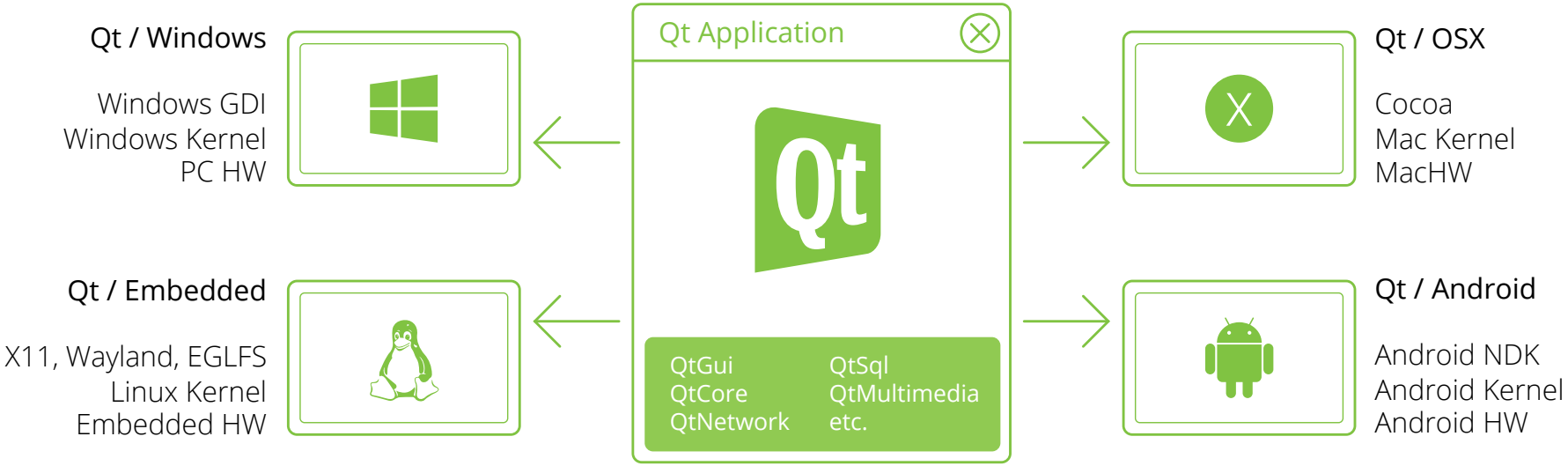
The Qt Quick World

The Qt Company



Qt Applications Are Native Applications

The Qt Company



Qt Quick Requirements

The Qt Company

- Platform must support OpenGL ES2
- Needs at least QtCore, QtGui, QtQml, and QtQuick modules
- Other modules can be used to add new features:
  - QtGraphicalEffects: add effects like blur, dropshadow...
  - Qt3D:3D programming in QML
  - QtMultimedia: audio and video items, camera
  - QtWebEngine: web view
  - ...

Qt Modules

The Qt framework is split into modules:

- Examples: QtCore, QtGui, QtWidgets, QtNetwork, QtMultimedia...
- Modules contain libraries, plugins and documentation.
- Libraries are linked to your applications
- Libraries group a set of common features (xml, dbus, network...)
- Qt Core is mandatory for all Qt applications

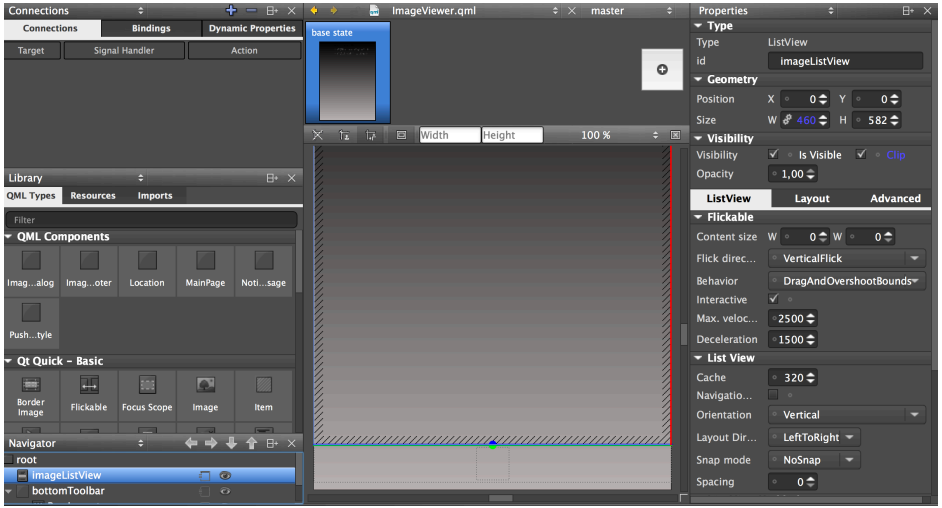
Meet Qt Quick

What is Qt Quick?

The Qt Company

A set of technologies including:

- Declarative markup language: QML
- Imperative Language: JavaScript
- Language runtime integrated with Qt
- C++ API for integration with Qt applications
- QtCreator IDE support for the QML language



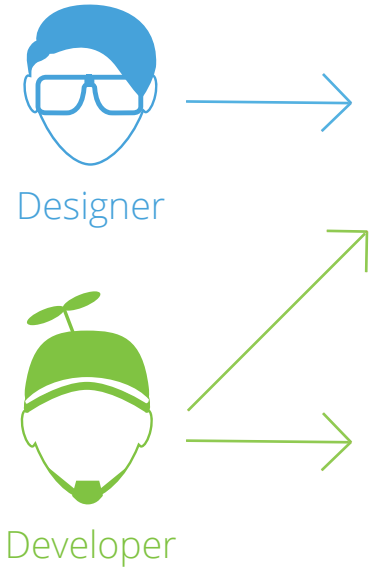
Philosophy of Qt Quick

The Qt Company

- Intuitive User Interfaces
- Design-Oriented
- Rapid Prototyping and Production
- Easy Deployment
- Enable designer and developers to work on the same sources

Rapid Workflow with Qt Quick

The Qt Company



Qt Quick

Declarative UI Design

Stunningly Fluent Modern User Interfaces, written with QML. Ideal for rapid UI prototyping.

Imperative Logic

Power of Cross-Platform Native Qt/C++

Core	Network	Sql	XML	Bluetooth	Positioning	NFC	Serial Port
Processes, Threads, IPC, Containers, I/O, Strings, Etc.	HTTP FTP SSL	SQL & Oracle Databases					

+ Direct Hardware Access

Concepts

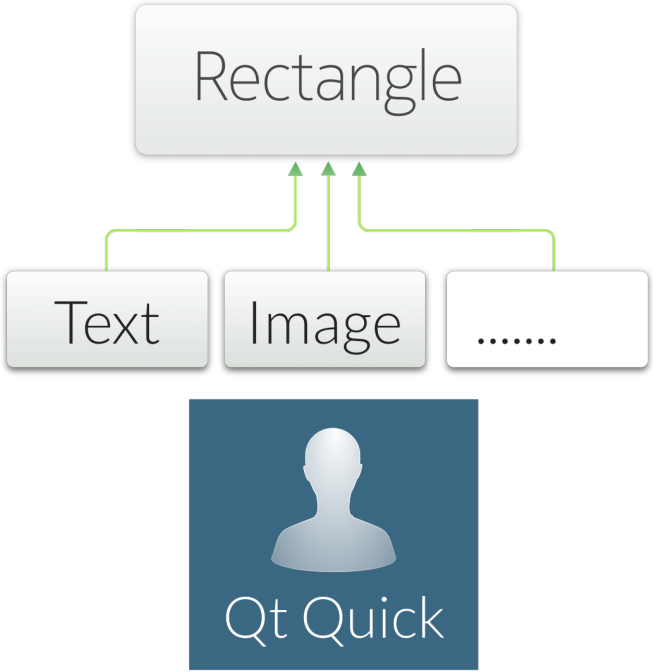
What Is QML?

The Qt Company

Declarative language for User Interface elements:

- Describes the user interface
  - What elements look like
  - How elements behave
- UI specified as tree of elements with properties

A Tree of Elements



- Let's start with an example...

Viewing an Example

The Qt Company

```
import QtQuick 2.4

Rectangle {
    width: 400;
    height: 400
    color: "lightblue"
}
```

- Locate the example: `rectangle.qml`
- Launch the QML runtime:

```
qmlscene rectangle.qml
```

Demo: qml-intro/ex-concepts/rectangle.qml

Elements

The Qt Company

- Elements are structures in the markup language
  - Represent visual and non-visual parts
- `Item` is the base type of visual elements
  - Not visible itself
  - Has a position, dimensions
  - Usually used to group visual elements
  - `Rectangle`, `Text`, `TextInput`,...
- Non-visual elements:
  - States, transitions,...
  - Models, paths,...
  - Gradients, timers, etc.
- Elements contain properties
  - Can also be extended with custom properties

See Documentation: QML Elements

Properties

Elements are described by properties:

- Simple name-value definitions
  - `width`, `height`, `color`,...
  - With default values
  - Each has a well-defined type
  - Separated by semicolons or line breaks
- Used for
  - Identifying elements (`id` property)
  - Customizing their appearance
  - Changing their behavior

Property Examples

The Qt Company

- Standard properties can be given values:

```
Text {
    text: "Hello world"
    height: 50
}
```

- Grouped properties keep related properties together:

```
Text {
    font.family: "Helvetica"
    font.pixelSize: 24
    // Prefferred syntax
    // font { family: "Helvetica"; pixelSize: 24 }
}
```

- Identity property gives the element a name:
  - Identifying elements (`id` property)
  - Customizing their appearance
  - Changing their behavior

```
Text {
    id: label
    text: "Hello world"
}
```

Property Examples

The Qt Company

- Attached properties are applied to elements:

```
TextInput {
    text: "Hello world"
    KeyNavigation.tab: nextInput
}
```

- `KeyNagivation.tab` is not a standard property of `TextInput`
- Is a standard property that is attached to elements

- Custom properties can be added to any element:

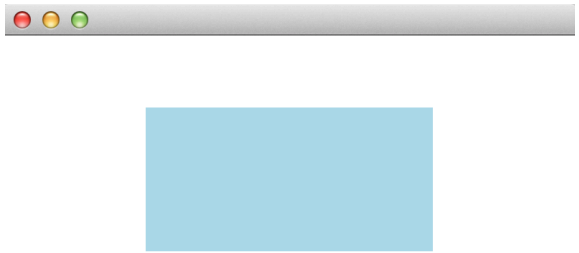
```
Rectangle {
    property real mass: 100.0
}

Circle {
    property real radius: 50.0
}
```

Binding Properties

```
Item {
    width: 400; height: 200
    Rectangle {
        x: 100; y: 50; width: height * 2; height: 100
        color: "lightblue"
    }
}
```

- Properties can contain expressions
  - See above: `width` is twice the `height`
- Not just initial assignments
- Expressions are re-evaluated when needed



Demo: qml-intro/ex-concepts/expressions.qml

Identifying Elements

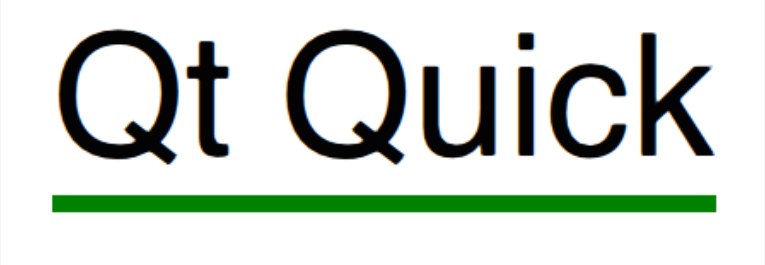
The `id` property defines an identity for an element

- Lets other elements refer to it
  - For relative alignment and positioning
  - To use its properties
  - To change its properties (e.g., for animation)
  - For re-use of common elements (e.g., gradients, images)
- Used to *create relationships* between elements

See Documentation: Property Binding

Using Identities

```
Item {
    width: 300; height: 115
    Text {
        id: title
        x: 50; y: 25 text: "Qt Quick"
        font.family: "Helvetica"; font.pixelSize: 50
    }
    Rectangle {
        x: 50; y: 75; height: 5
        width: title.width
        color: "green"
    }
}
```

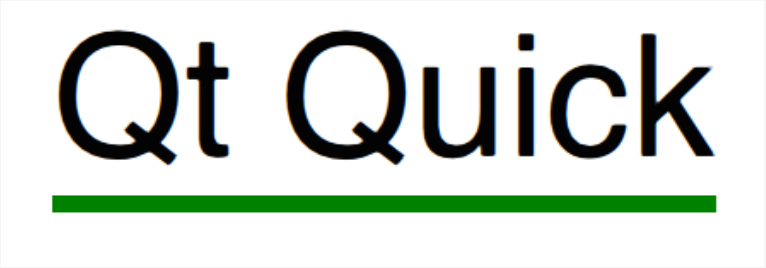


Demo: qml-intro/ex-concepts/identity.qml

Viewing an Example

The Qt Company

```
Text {
    id: title
    x: 50; y: 25 text: "Qt Quick"
    font.family: "Helvetica"; font.pixelSize: 50
}
Rectangle {
    x: 50; y: 75; height: 5
    width: title.width
    color: "green"
}
```



- Property `Text` element has the identity, title
- Property `width` of `Rectangle` bound to `width` of title

Basic Types

The Qt Company

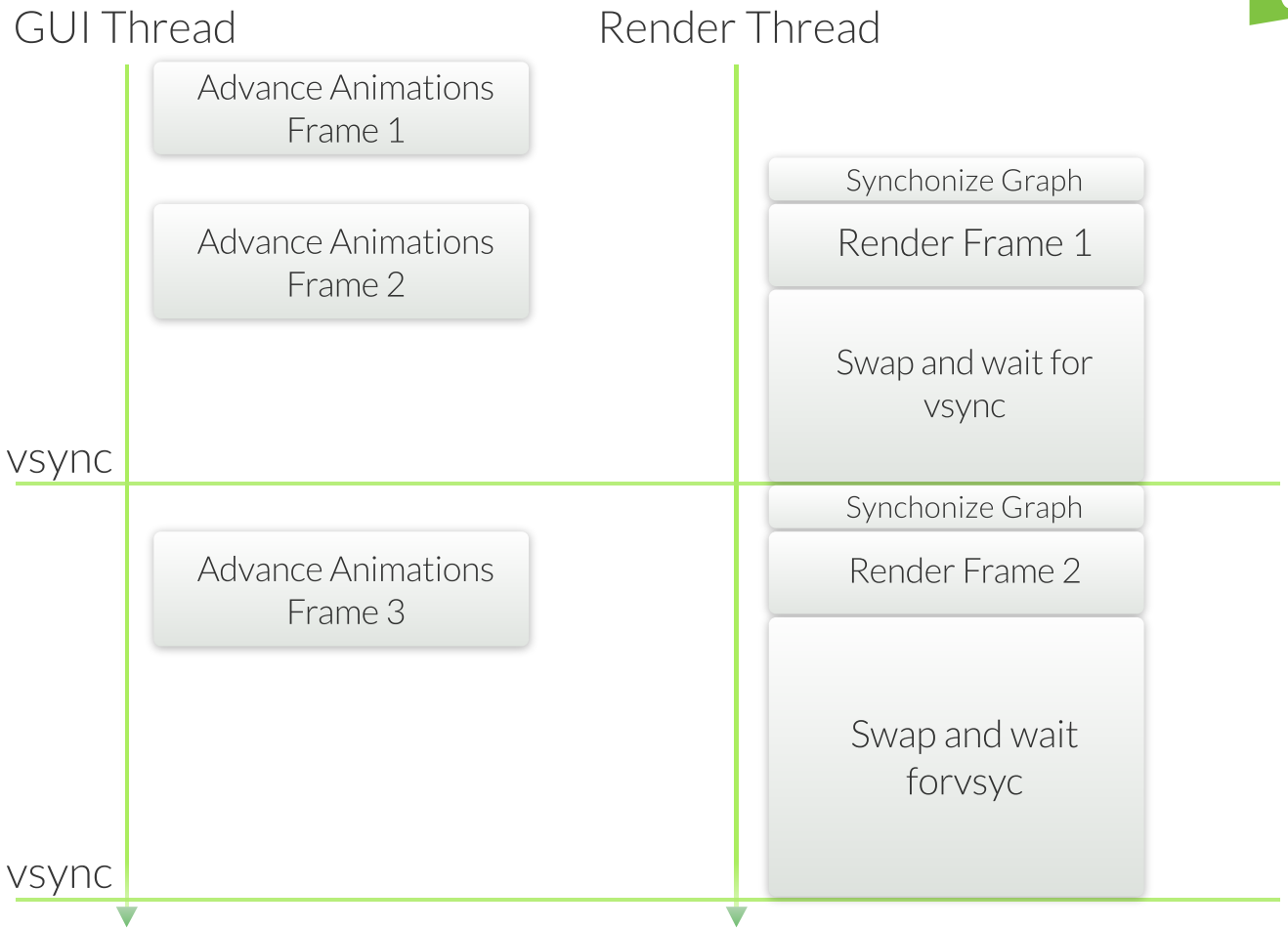
Property values can have different types:

- Numbers (int and real): 400 and 1.5
- Boolean values: `true` and `false`
- Strings: `"HelloQt"`
- Constants: `AlignLeft`

- Lists:[...]
  - One item lists do not need brackets
- Scripts:
  - Included directly in property definitions
- Other types:
  - colors, dates, rects, sizes, 3Dvectors,...
  - Usually created using constructors

See Documentation: QML Types

Behind the Scene



Summary

The Qt Company

- QML defines user interfaces using elements and properties
  - Elements are the structures in QML source code
  - Items are visual elements
- Standard elements contain properties and methods
  - Properties can be changed from their default values
  - Property values can be expressions
  - `Id` properties give identities to elements
- Properties are bound together
  - When a property changes, the properties that reference it are updated
- Some standard elements define methods
- A range of built-in types is provided

Questions

The Qt Company

- How do you load a QML module?
- What is the difference between `Rectangle` and `width`?
- How would you create an element with an identity?
- What syntax do you use to refer to a property of another element?

Lab – Items

The image on the right shows two items and two child items inside a 400 × 400 rectangle.

1. Recreate the scene using Rectangle items.
2. Can items overlap? Experiment by moving the light blue or green rectangles.
3. Can child items be displayed outside their parents? Experiment by giving one of the child items negative coordinates.