

# Introduction à Qt - QML

Lionel Duboeuf – Code d'Armor - Avril 2019

# Qt - Kesako ?

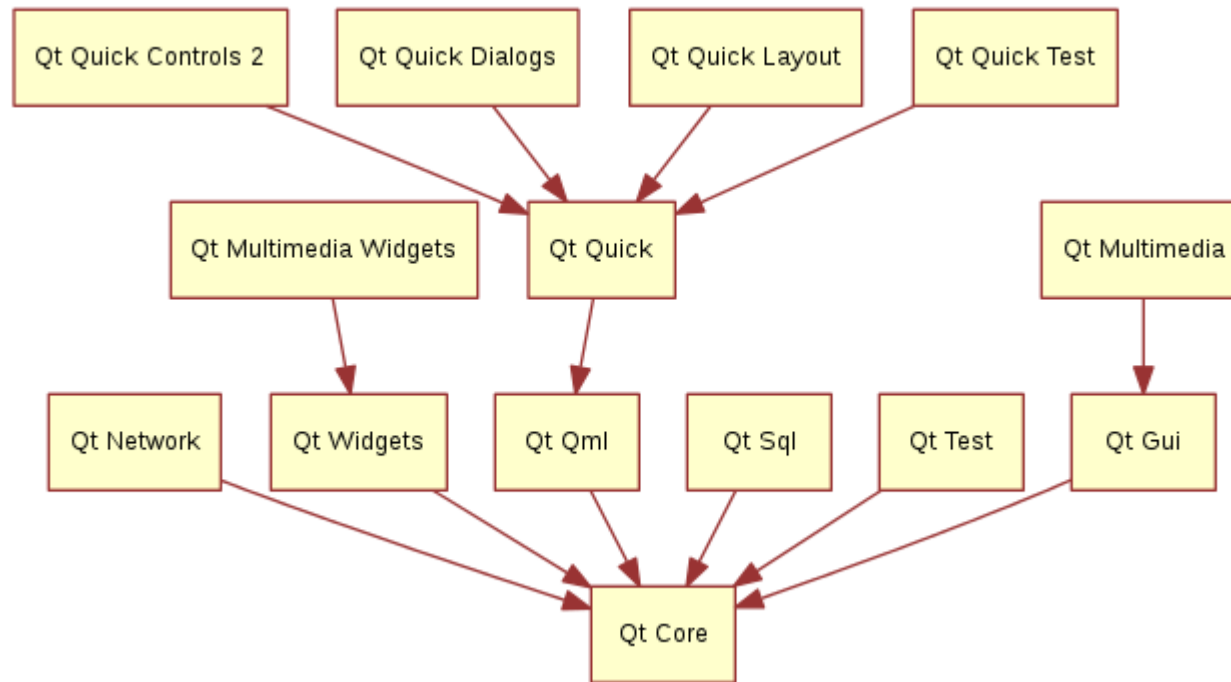
- **Qt (“cute”) = framework multi plateformes**
- **Ecrit en c++**
- **Initié en 1991 par Haavard Nord and Eirik Chambe-Eng en Norvège**
- **Trolltech → Nokia → Digia → The Qt Company**
- **Licence commerciale et Open Source ( GPL et LGPL v3)**
- **Présent dans l’embarqué, applications Desktop et Mobile. Environnements de bureau ( ex: KDE, Unity , LxQt, ... )**
- **Version actuelle: 5.12**

# Qt - Ecosystème

- **IDE = QtCreator**
- **Prototypage = QmlScene**
- **Live reloading: qmllivebench**
- **Build = Qmake / Cmake**
- **Tests**
- **Internationalisation: Linguist**
- **Extensible en C++**
- **Architectures possible: C++ ↔ QML/QtWidget (traditionnelle ), Python ↔ QML/QtWidget, Javascript ↔ QML, ...**
-

# Qt - Modules

- **Modules essentiels:**



source

# Qt - Qt Modeling Language (QML)

- **Language déclaratif ( proche du css, json )**
- **Inclus un moteur Javascript simplifié ( EcmaScript 7 avec la 5.12 LTS )**
- **QtQuick :**
  - ensemble de composants visuels élémentaires et de fonctions pour l'animation, les i/o, etc...)
- **QtQuick Controls 2**
  - Ensemble de composants étendus ( boutons, listes, champs de saisie, navigation avec onglets, menus, styles etc... )
    - > Demo Gallery App ( disponible dans QtCreator )

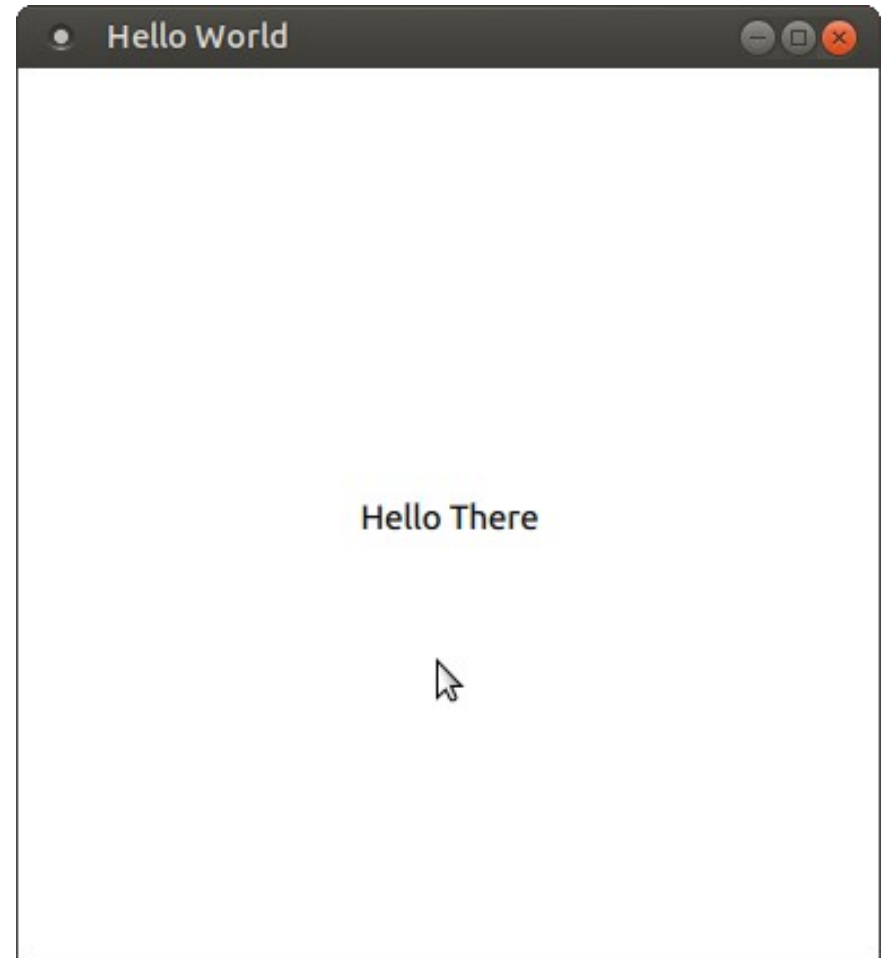
# QtQuick - Hello World - Structure



- **“.pro” = directives de compilation**
- **“main.cpp” = point d’entrée de l’application (chargement du moteur, librairies, logique et lancement de la GUI )**
- **“.qrc” = fichier contenant toutes les ressources du projet**
- **“.qml” = QML :)**

# QtQuick - Hello World

```
< > qml main.qml
1 import QtQuick 2.9
2 import QtQuick.Window 2.2
3
4 Window {
5     visible: true
6     width: 640
7     height: 480
8     title: qsTr("Hello World")
9
10
11 Text{
12     anchors.centerIn: parent
13     text: qsTr("Hello There")
14 }
15
16
17
```



# QML - Syntaxe, propriétés, bindings, signaux

```
1  import QtQuick 2.0
2
3  // The root element is the Rectangle
4  Rectangle {
5      // name this element root
6      id: root
7
8      // properties: <name>: <value>
9      width: 120;
10
11     //property bindings: here height will changed according to width changed
12     height: width * 2
13
14     // custom property: can be functions, QML types, var, int, string, etc...
15     property int times: 24
16
17     // property alias ( bind to a nested property )
18     property alias name: txt.text
19
20
21     // signal handler for property changes, call javascript function
22     onHeightChanged: console.log('height:', height)
23
24     // color property
25     color: "#4A4A4A"
26
27     // Declare a nested element (child of root)
28     Text{
29         id: txt
30         //js expression
31         color: focus ? "red":"black"
32
33         anchors.centerIn: parent
34     }
35
36 }
37
38
```



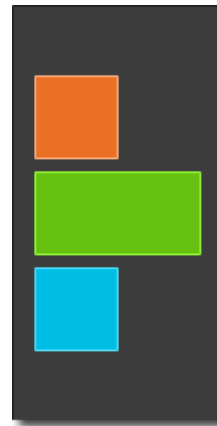
# QML - Notion de composants

```
Counter.qml
1 import QtQuick 2.0
2 import QtQuick.Controls 2.2
3
4 Rectangle{
5
6     id: root
7     width:120
8     height:120
9
10    property int count:0
11    property int maxCount: 10
12
13    signal finished
14
15    Column {
16        Text {
17            text: root.count
18            color: "blue"
19        }
20
21        Button{
22            text: qsTr("increment")
23            onClicked: {
24
25                root.count++
26                if (root.count >= root.maxCount ){
27                    finished()
28                }
29
30            }
31        }
32    }
33 }
34
35
36
37
38 }
39
```

```
main.qml
1 import QtQuick 2.9
2 import QtQuick.Window 2.2
3
4 Window {
5     visible: true
6     width: 640
7     height: 480
8     title: qsTr("Hello World")
9
10    Counter {
11        onFinish: console.log("counter finished")
12    }
13
14
15
16 }
```

# QML - Positionnement - Layout

- Utilisation de conteneurs: Row, Column, Grid:

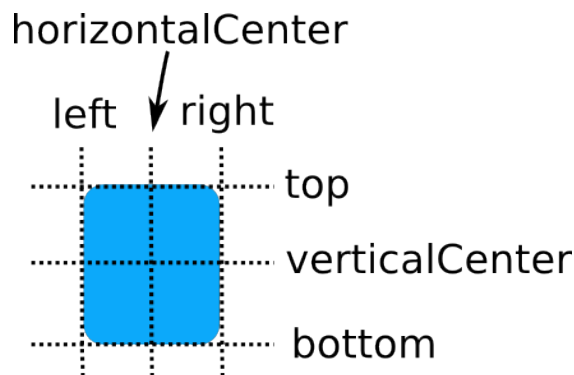


## Exemple:

```
Row {  
    spacing: 10  
    Rectangle { width:50; height:50; color: "blue"}  
    Rectangle { width:50; height:50; color: "green"}  
    Rectangle { width:50; height:50; color: "blue"}  
}
```

# QML - Positionnement - Layout

- **anchors:** positionnement par rapport à un parent, un sibling ou autre.



Exemples d'utilisation:

`anchors.fill : parent // meme taille que le parent`  
`anchors.centerIn: parent // au milieu du parent`  
`anchors.left: xxx.left // on calle a gauche de xxx`  
`anchors.margins: 12 // marge de 12`

- **Layout: ( Module QtQuick.Layout )**
  - Facilite le positionnement. Utilisation de RowLayout et ColumnLayout par ex. + plusieurs propriétés étendues

# Scripting

- Utilisation de Javascript dans des fichiers externes, à l'intérieur du QML ou en tant que librairie (singleton)

```
Button {  
    width: 200  
    height: 300  
    property bool checked: false  
    text: "Click to toggle"  
  
    // JS function  
    function doToggle() {  
        checked = !checked  
    }  
  
    onTriggered: {  
        // this is also JavaScript  
        doToggle();  
        console.log('checked: ' + checked)  
    }  
}
```

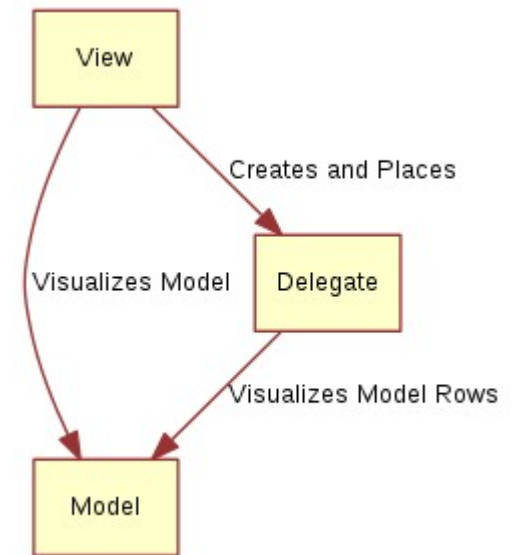
```
Util.js  
1 function sayHello() {  
2     return "kikou"  
3 }  
4
```

```
DemoJs.qml  
1 import QtQuick 2.0  
2 import QtQuick.Controls 2.0  
3  
4 import "Util.js" as Util  
5  
6 Item {  
7  
8     Button {  
9         width: 200  
10        height: width*2  
11  
12        onClicked: {  
13            // this is JavaScript  
14            console.log(Util.sayHello())  
15        }  
16    }  
17 }  
18 }  
19
```

# Les listes de données

## Architecture décomposée en “Model View Delegate”

- Model: Les données. Peut être un nombre, un tableau de primitives ou une liste d'objets
- View: Le conteneur ( exemple: ListView )
- Delegate: Le décorateur, défini comment doit s'afficher chacune des données

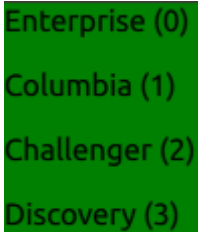


# Les listes de données

```
Column {
  anchors.fill: parent
  spacing:4
  ListView{
    anchors.fill: parent
    model: ["Enterprise", "Columbia", "Challenger", "Discovery"]

    delegate: Rectangle {
      width: 100
      height: 32
      color: "green"

      Text {
        text: modelData + ' (' + index + ')'
      }
    }
  }
}
```



Enterprise (0)  
Columbia (1)  
Challenger (2)  
Discovery (3)

```
ListView{
  anchors.fill: parent
  model: ListModel {
    id: elements
    ListElement {name:"Enterprise"}
    ListElement {name:"Columbia"}
    ListElement {name:"Challenger"}
    ListElement {name:"Discovery"}
  }
  delegate: Rectangle {
    width: 100
    height: 32
    color: "green"
    Text { text: name + ' (' + index + ')' }
  }
}
```



# Animations

- **Animation autonome, lancée via la propriété “running “ ou start()**
- **Animation lors du chargement de l'élément (ici animation sur l'opacité ):**

```
NumberAnimation on opacity {  
    from:0; to:1  
    duration:2000  
}
```

- **Lors d'un changement de valeur d'une propriété:**

```
Behavior on width {  
    NumberAnimation { duration: 1000 }  
}
```

# Animations

- Regroupement d'animations : execution parallèle (ParallelAnimation ) ou séquentielle (SequentialAnimation )
- Possibilité de modifier les courbes d'interpolation "Easing Curves"
- → Demo Animations

```
Rectangle {
    id: example
    //anchors.horizontalCenter: pa
    width: 100; height: 100
    color: "blue"
    ParallelAnimation{
        running: myMouse.pressed
        loops: Animation.Infinite
        NumberAnimation {
            target: example
            property: "x"
            from: 0; to: 100
            duration:2000
        }

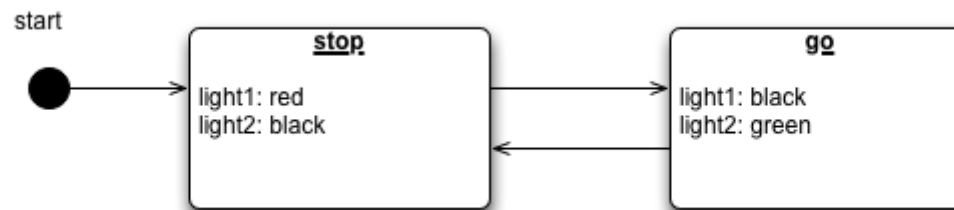
        ColorAnimation {
            target: example
            property: "color"
            from: "blue"
            to: "black"
            duration: 2000
        }

        RotationAnimation{
            target: example
            from:0
            to:360
            duration:1000
        }
    }
}
```



# Etats et transitions

- **Etats: permet d'appliquer des comportements ou/et des changements de propriétés.**
  - Exemple pour des feux de circulation:



```
states: [  
  State {  
    name: "stop"  
    PropertyChanges { target: light1; color: root.red }  
    PropertyChanges { target: light2; color: root.black }  
  },  
  State {  
    name: "go"  
    PropertyChanges { target: light1; color: root.black }  
    PropertyChanges { target: light2; color: root.green }  
  }  
]
```

```
MouseArea {  
  anchors.fill: parent  
  onClicked: parent.state = (parent.state == "stop"? "go" : "stop")  
}
```

# Etats et transitions

- **Transitions: Permet d'appliquer des transformations ou actions lors d'un passage d'un état à un autre**

```
transitions: [  
    Transition {  
        from: "stop"; to: "go"  
        ColorAnimation { target: light1; properties: "color"; duration: 2000 }  
        ColorAnimation { target: light2; properties: "color"; duration: 2000 }  
    }  
]
```

# LocalStorage

- LocalStorage = bdd SQLite

```
import QtQuick 2.5
import QtQuick.LocalStorage 2.0

Item {
    Component.onCompleted: {
        var db = LocalStorage.openDatabaseSync("MyExample", "1.0", "Example database", 10000);
        db.transaction( function(tx) {
            var result = tx.executeSql('select * from notes');
            for(var i = 0; i < result.rows.length; i++) {
                print(result.rows[i].text);
            }
        });
    }
}
```

- Utilisation de l'objet javascript xhr

```
function getAll(onSuccess, onError){  
  
    var http = new XMLHttpRequest()  
    var url = Config.API_URL;  
    http.open("GET", url, true);  
  
    // Send the proper header information along with the request  
    http.setRequestHeader("Content-type", "application/json");  
    http.setRequestHeader("TOKEN", Config.API_KEY);  
  
    http.onreadystatechange = function() { // Call a function when the state changes.  
        if (http.readyState == 4) {  
            if (http.status == 200) {  
  
                //onLineScores.clear()  
                var scores = []  
                var recordToHighlight = 0  
                var fetchScores = JSON.parse(http.responseText)  
                for(var i = 0; i < fetchScores.length; i++){  
  
                    var record = fetchScores[i]  
                    scores.push(record)  
  
                }  
  
                onSuccess(scores)  
  
            } else {  
                onError(http.status)  
            }  
        }  
    }  
    http.send();  
}
```

# Encore plus

- **Module Settings: Stockage en local des données de configuration de l'application**
- **Qt WebEngine: Moteur de rendu Web ( Chromium )**
- **Shaders, Particles**
- **Theming de l'application: Module Material par ex.**

# Encore plus...

- **Qt 3D** A set of APIs to make 3D graphics programming easy and declarative.
- **Qt Bluetooth** C++ and QML APIs for platforms using Bluetooth wireless technology.
- **Qt Canvas 3D** Enables OpenGL-like 3D drawing calls from Qt Quick applications using JavaScript.
- **Qt Graphical Effects** Graphical effects for use with Qt Quick 2.
- **Qt Location** Displays map, navigation, and place content in a QML application.
- **Qt Network Authorization** Provides support for OAuth-based authorization to online services.
- **Qt Positioning** Provides access to position, satellite and area monitoring classes.
- **Qt Purchasing** Enables in-app purchase of products in Qt applications. (Only for Android, iOS and MacOS).
- **Qt Sensors** Provides access to sensors and motion gesture recognition.
- **Qt Wayland Compositor** Provides a framework to develop a Wayland compositor. (Only for Linux).
- **Qt Virtual Keyboard** A framework for implementing different input methods as well as a QML virtual keyboard. Supports localized keyboard layouts and custom visual themes.

# Retour d'expérience

- **Courbe d'apprentissage un peu près identique à l'approche HTML/CSS pour ce qui est du Layout , positionnements**
- **Grande souplesse de conception = top, mais on peut vite faire aussi n'importe quoi... :)**
- **Documentation très complète mais parfois des informations importantes sont perdues au milieu de la doc.**
- **Quelques hackings nécessaires pour que ça tourne bien coté mobile.**

# Resources et credits

- <https://doc.qt.io/>
- <http://qmlbook.github.io>
- Internet...





**merci!**