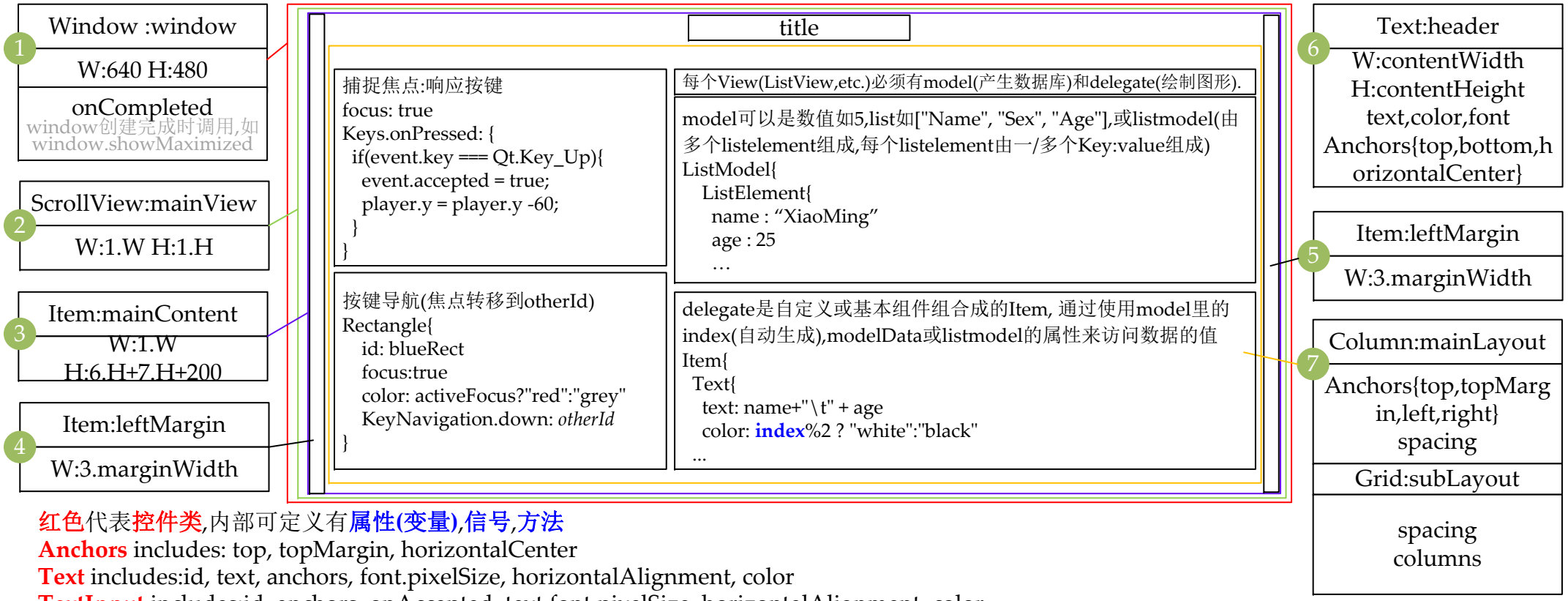
QML支持的函数:
1. Math.round(5*1.3),Qt.quit(),qsTr("myButton"),console.log("hello"),Screen.width,
2. Image类中: 使用变量progress查询加载进度; 变量Image.status(值为ready表示图片已加载);

一般顶层的Rectangle的ID为root;
.qmlproject可以设置QML文件,JS文件及图片文件的目录



红色代表控件类,内部可定义有属性(变量),信号,方法
Anchors includes: top, topMargin, horizontalCenter
Text includes:id, text, anchors, font.pixelSize, horizontalAlignment, color
TextInput includes:id, anchors, onAccepted, text,font.pixelSize, horizontalAlignment, color
MouseArea includes:id, anchors, onClicked (,hoverEntered)(,onEntered)(,onExited)
Rectangle includes:id,width,height,color(,x)(,y)(,anchors)(border.color)(border.width)(radius)(,**Text**)(,**MouseArea**)
Image includes:id,source,sourceSize.height,sourceSize.width,anchors,onProgressChanged,onStatusChanged(,**Text**)(,**MouseArea**)
AnimatedImage includes:id,source,(x),(y),anchors,onProgressChanged,onStatusChanged(,**Text**)(,**MouseArea**)
Flickable(滚动条) includes:id,width,height(滚动区域), contentWidth, contentHeight(要滚动的内容), contentY(当前显示内容相对于Flickable区域左上角的Y offset值, 垂直滚动的设置: Math.min(contentHeight-height, Math.max(0,player.y-height/2))), boundsBehavior: Flickable.StopAtBounds, interactive: true
自定义Item includes: 先定义属性(`property bool/string/int/double [属性名]:初始值`),信号(`signal buttonClicked`,使用时直接`buttonClicked()`即触发信号), 再实例化内部需要的其他控件.
使用自定义Item时: 控件类名(=自定义Item所在文件名){… 设置属性…,… 实现槽函数onButtonClicked:{console.log("I'm slot")}}
Repeater类:包含一个model和一个delegate, model常为数值(i.e.5),delegate可以是任意Item类. Repeater一般被包含在Row,Column,Grid类的实例中，生成一行/列/框相类似的Item.

Item的子类:AnimatedSprite, BorderImage, Canvas, Column, ColumnLayout, Flickable, Flipable, Flow, FocusScope, Grid, GridLayout, Image, Loader, MouseArea, MultiPointTouchArea, ParticlePainter, PathView, PinchArea, Rectangle, Repeater, Row, RowLayout, ShaderEffect, ShaderEffectSource, Shape, SignalSpy, SpriteSequence, StackLayout, TestCase, Text, TextEdit, and TextInput

第3种动画的方式:使用状态机分别写状态和跳转
* State中实现AnchorChanges, PropertyChanges等的赋值
* Transition中实现状态跳转时发生的动画,如果只有一State,则只有一个Transition
* 跳转条件:在需要触发处对**rect2**.state赋值即可

```
Rectangle{
id: rect2
width: Screen.width/2
height: Screen.height/12
color:"transparent"
radius: 20
states: [
  State {
  name: "ENTERED"
      PropertyChanges {
        target: rect2
        color:"orange"
      }
  },
  State {
    name: "EXITED"
    PropertyChanges {
      target: rect2
      color:"transparent"
    }
  }
]
transitions: [
  Transition {
    from: "EXITED"
    to: "ENTERED"
    ColorAnimation {target:rect2;duration: 1200}
  },
  Transition {
    from: "ENTERED"
    to: "EXITED"
    ColorAnimation{ target: rect2; duration:1200}
  }
]
}
```

动画3种

第1种动画实现的方式
* Behavior on property:当属性的值发生变化时,将会触发下面的动作

```
Behavior on scale{
  NumberAnimation{
    //持续800ms
    duration: 800
    easing.type:Easing.OutBounce
  }
}
```

第2种动画实现的方式
* PropertyAnimation是用来为属性提供动画的最基本的动画元素，可以用来为real、int、color、rect、point、size和vector3d等属性设置动画，
* NumberAnimation、colorAnimation、RotationAnimation和Vector3dAnimation等元素继承。
* NumberAnimation对real和int属性提供了更高效的实现;
* Vector3dAnimation对vector3d属性提供了更高效的支持;
* ColorAnimation和RotationAnimation分别对color和rotation属性变化动画提供了特定的属性支持。
* 在需要触发时进行**scaleAnimation.**start()即可

```
    PropertyAnimation{
        id:scaleAnimation
        target: imageOfElement
        property: "scale"
        from:0
        to:1
        duration: 1000
        easing.type: Easing.OutBack
    }

    //NumberAnimation修改某属性的数值
    NumberAnimation{
        id:opacityAnimation
        target: imageOfElement
        property: "opacity"
        from:0
        to:1
        duration: 2800
        easing.type: Easing.OutBack

    }
```

Loader 加载页面(*.qml)
Timer 定时器
并行动画ParallelAnimation
顺序动画SequentialAnimation

XmlListModel使用互联网API生成线性Model

使用XmlListModel使用互联网API生成线性Model
(属性`title`和`pubDate`可直接被View访问)

```
import QtQuick 2.0
import QtQuick.XmlListModel 2.0
XmlListModel {
    id: xmlModel
    source: "http://www.mysite.com/feed.xml"
    query: "/rss/channel/item"
    XmlRole { name: "title"; query: "title/string()" }
    XmlRole { name: "pubDate"; query: "pubDate/string()" }
}
```

SystemPalette 使用系统自带配色

系统窗口的渐变色gradient: Gradient{ GradientStop{position:0.9;color:Qt.darker(palette.window, 1.8)}