**QML之QtCharts模块**

订阅专栏

Qt有QtCharts模块，主要针对能够快速的绘制出各种类型的表格,QtCharts主要包含的表格有:  
1.PieSeries([饼状图](https://so.csdn.net/so/search?q=%E9%A5%BC%E7%8A%B6%E5%9B%BE&spm=1001.2101.3001.7020))  
2.LineSeries([折线图](https://so.csdn.net/so/search?q=%E6%8A%98%E7%BA%BF%E5%9B%BE&spm=1001.2101.3001.7020))  
3.SplineSeries([曲线图](https://so.csdn.net/so/search?q=%E6%9B%B2%E7%BA%BF%E5%9B%BE&spm=1001.2101.3001.7020))  
4.ScatterSeries([散点图](https://so.csdn.net/so/search?q=%E6%95%A3%E7%82%B9%E5%9B%BE&spm=1001.2101.3001.7020))  
5.BarSeries(条状图)  
6.StackedBarSeries(层叠条状图)  
7.PercentBarSeries(百分比条状图)  
8.HorizontalBarSeries(水平条状图)  
9.HorizontalStackedBarSeries(水平层叠条状图)  
10.HorizontalPercentBarSeries(水平百分比条状图)  
下面一一做些小例子:

1.PieSeries(饼状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

id:chart

title:"对各编程语言掌握程度"

anchors.fill:parent

legend.alignment:Qt.AlignBottom //图表样例底部居中

antialiasing:true //是否抗锯齿

PieSeries {

id:pieSeries

PieSlice {label:"Go";value:13.5;color:"red"}

PieSlice {label:"Java"; value:10.9;color:"yellow"}

PieSlice {label:"Matlab"; value:8.6;color:"blue"}

PieSlice {label:"PHP"; value:8.2;color:"cyan"}

PieSlice {label:"JS"; value:6.8;color:"black"}

}

}

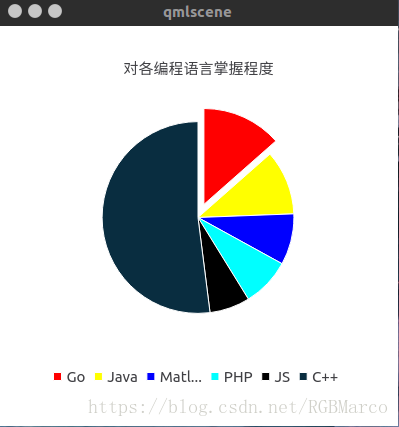
Component.onCompleted: {

pieSeries.append("C++",52.0) //动态增加Label为”C++"的元素

pieSeries.find("Go").exploded = true; //使带“Go"Lable的元素暴露

}

}



2.LineSeries(折线图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:300

height:300

ChartView {

title:"折线图"

id:chart

anchors.fill:parent

antialiasing:true

legend.alignment:Qt.AlignBottom

LineSeries {

name:"学生"

id:line

XYPoint {x:0;y:0}

XYPoint {x:1.1;y:2.1}

XYPoint {x:1.9;y:3.3}

XYPoint {x:2.1;y:2.1}

XYPoint {x:2.9;y:4.9}

XYPoint {x:3.4;y:3.0}

XYPoint {x:4.1;y:3.3}

}

}

Component.onCompleted: {

}

}



3.SplineSeries(曲线图)

import QtQuick 2.0

import QtCharts 2.0

Item {

id:root

width:400

height:400

ChartView {

title:"曲线图"

antialiasing:true

legend.alignment:Qt.AlignTop

anchors.fill:parent

SplineSeries {

name:"学生"

XYPoint{x:0.0;y:0.0}

XYPoint{x:0.12;y:0.4}

XYPoint{x:0.3;y:0.7}

XYPoint{x:0.25;y:0.66}

XYPoint{x:0.4;y:0.83}

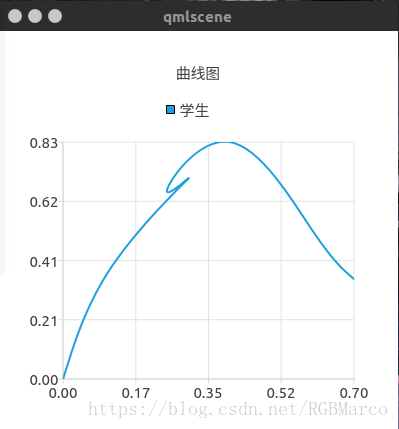
XYPoint{x:0.6;y:0.52}

XYPoint{x:0.7;y:0.35}

}

}

}



4.ScatterSeries(散点图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

title:"散点图"

anchors.fill:parent

antialiasing:true

ValueAxis {

id:valueAx

min:1998

tickCount:9

max:2006

labelFormat:"%d"

}

ScatterSeries {

axisX:valueAx

id:scatter1

name:"散点分布"

XYPoint {x:1998;y:2}

XYPoint {x:1999;y:1.5}

XYPoint {x:2000;y:2.3}

XYPoint {x:2001;y:2.2}

XYPoint {x:2002;y:2.4}

XYPoint {x:2003;y:3.1}

XYPoint {x:2004;y:1.0}

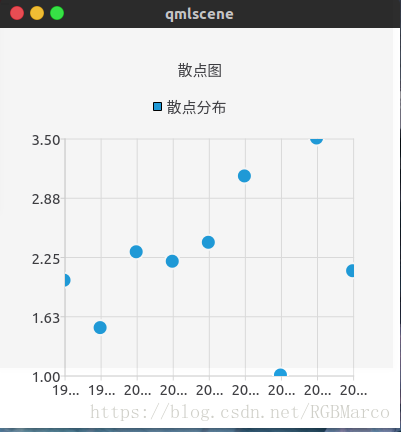
XYPoint {x:2005;y:3.5}

XYPoint {x:2006;y:2.1}

}

}

}



5.AreaSeries(区域图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

title:"区域图"

antialiasing:true

legend.alignment:Qt.AlignTop

anchors.fill:parent

ValueAxis {

id:valueAxis

min:2000

max:2011

tickCount:12

labelFormat:"%.0f"

}

AreaSeries {

name:"俄罗斯"

axisX:valueAxis

upperSeries:LineSeries {

XYPoint {x:2000;y:1}

XYPoint {x:2001;y:1}

XYPoint {x:2002;y:1}

XYPoint {x:2003;y:1}

XYPoint {x:2004;y:0}

XYPoint {x:2005;y:0}

XYPoint {x:2006;y:1}

XYPoint {x:2007;y:1}

XYPoint {x:2008;y:4}

XYPoint {x:2009;y:3}

XYPoint {x:2010;y:2}

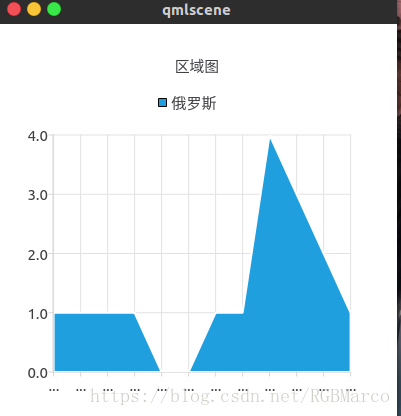
XYPoint {x:2011;y:1}

}

}

}

}



完整版

|  |  |
| --- | --- |
| import QtQuick 2.15  import QtQuick.Window 2.15  import QtCharts 2.5  import QtQuick.Controls 2.5  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Hello World")  //AreaSeries  ChartView {  title: "NHL All-Star Team Players"  anchors.fill: *parent*  antialiasing: true  // Define x-axis to be used with the series instead of default one  ValueAxis {  id: *valueAxis*  min: 2000  max: 2011  tickCount: 12  labelFormat: "%.0f"  }  AreaSeries {  name: "Russian"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2000; y: 1 }  XYPoint { x: 2001; y: 1 }  XYPoint { x: 2002; y: 1 }  XYPoint { x: 2003; y: 1 }  XYPoint { x: 2004; y: 1 }  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 1 }  XYPoint { x: 2007; y: 1 }  XYPoint { x: 2008; y: 4 }  XYPoint { x: 2009; y: 3 }  XYPoint { x: 2010; y: 2 }  XYPoint { x: 2011; y: 1 }  }  }  AreaSeries {  name: "Swidesh"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2000; y: 1 }  XYPoint { x: 2001; y: 1 }  XYPoint { x: 2002; y: 3 }  XYPoint { x: 2003; y: 3 }  XYPoint { x: 2004; y: 2 }  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 2 }  XYPoint { x: 2007; y: 1 }  XYPoint { x: 2008; y: 2 }  XYPoint { x: 2009; y: 1 }  XYPoint { x: 2010; y: 3 }  XYPoint { x: 2011; y: 3 }  }  }  AreaSeries {  name: "Finnish"  axisX: *valueAxis*  upperSeries: LineSeries {  XYPoint { x: 2005; y: 0 }  XYPoint { x: 2006; y: 1 }  XYPoint { x: 2007; y: 0 }  XYPoint { x: 2010; y: 0 }  XYPoint { x: 2011; y: 1 }  }  }  }  } |  |

6.BarSeries(条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

title:"条状图"

anchors.fill:parent

antialiasing:true

legend.alignment:Qt.AlignBottom

BarSeries {

axisX:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

id:barseries

BarSet {label:"Bob";values:[2,2,3,4,5,6]}

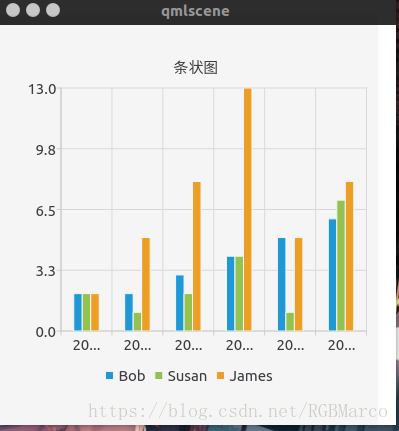
BarSet {label:"Susan";values:[2,1,2,4,1,7]}

BarSet {label:"James";values:[2,5,8,13,5,8]}

}

}

}



7.StackedBarSeries(层叠条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

id:chart

title:"层叠条状图"

antialiasing:true

legend.alignment:Qt.AlignBottom

anchors.fill:parent

StackedBarSeries {

axisX:BarCategoryAxis {categories:["2007","2008","2009","2010","2011","2012"]}

BarSet {label:"Bob";values:[2,2,3,4,5,6]}

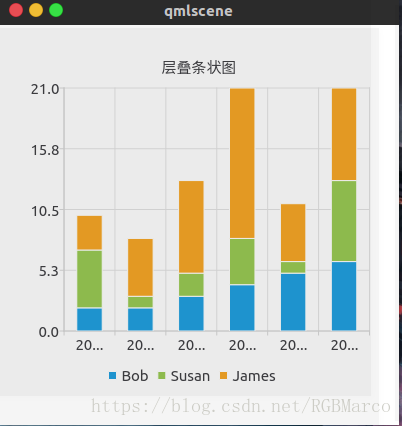
BarSet {label:"Susan";values:[5,1,2,4,1,7]}

BarSet {label:"James";values:[3,5,8,13,5,8]}

}

}

}`



8.PercentBarSeries(百分比条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

id:chart

anchors.fill:parent

antialiasing:true

title:"百分比条状图"

legend.alignment:Qt.AlignTop

PercentBarSeries {

axisX:BarCategoryAxis{categories:{["2007","2008","2009","2010","2011","2012"]}}

BarSet{label:"Bob";values:[2,2,3,4,5,6]}

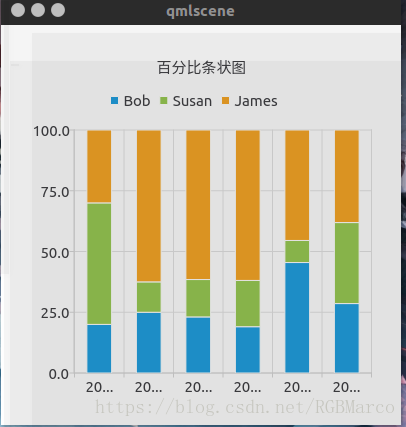
BarSet{label:"Susan";values:[5,1,2,4,1,7]}

BarSet{label:"James";values:[3,5,8,13,5,8]}

}

}

}



9.HorizontalBarSeries(水平条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

title:"水平条状图"

anchors.fill:parent

antialiasing:true

legend.alignment:Qt.AlignBottom

HorizontalBarSeries {

axisY:BarCategoryAxis{categories:{["2007","2008","2009","2010","2011","2012"]}}

BarSet{label:"Bob";values:[2,2,3,4,5,6]}

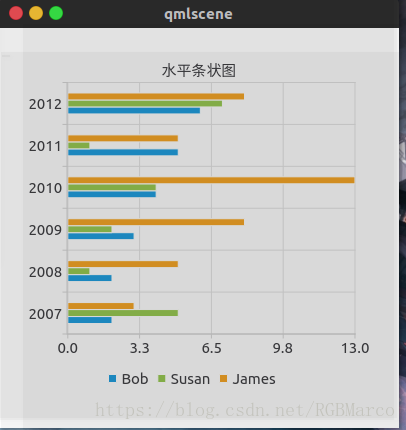
BarSet{label:"Susan";values:[5,1,2,4,1,7]}

BarSet{label:"James";values:[3,5,8,13,5,8]}

}

}

}



10.HorizontalStackedBarSeries(水平层叠条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

id:chart

title:"水平层叠图"

anchors.fill:parent

antialiasing:true

legend.alignment:Qt.AlignBottom

HorizontalStackedBarSeries {

axisX:BarCategoryAxis{categories:{["2007","2008","2009","2010","2011","2012"]}}

BarSet{label:"Bob";values:[2,2,3,4,5,6]}

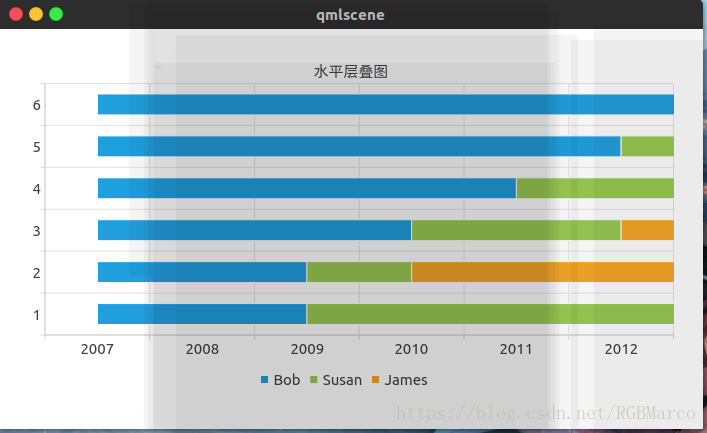
BarSet{label:"Susan";values:[5,1,2,4,1,7]}

BarSet{label:"James";values:[3,5,8,13,5,8]}

}

}

}



11.HorizontalPercentBarSeries(水平百分比条状图)

import QtQuick 2.0

import QtCharts 2.0

Item {

width:400

height:400

ChartView {

title:"层叠水平百分比条状图"

anchors.fill:parent

antialiasing:true

legend.alignment:Qt.AlignBottom

HorizontalPercentBarSeries {

axisY:BarCategoryAxis{categories:{["2007","2008","2009","2010","2011","2012"]}}

BarSet{label:"Bob";values:[2,2,3,4,5,6]}

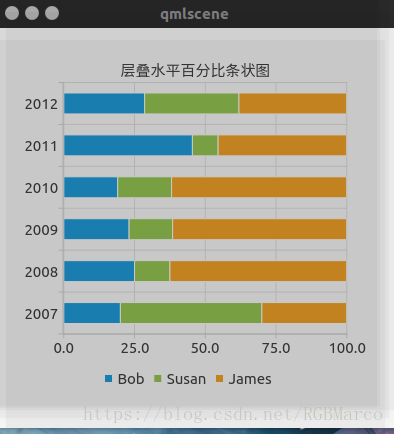
BarSet{label:"Susan";values:[5,1,2,4,1,7]}

BarSet{label:"James";values:[3,5,8,13,5,8]}

}

}

}



综合:

“import QtQuick 2.0  
import QtCharts 2.0

Item {  
width:800  
height:700  
Row {  
id:row1  
anchors.left:parent.left  
anchors.leftMargin:4  
ChartView {  
id:pie  
antialiasing:true  
title:”饼状图”  
width:200  
height:200  
legend.alignment:Qt.AlignTop  
PieSeries {  
holeSize:0.5  
PieSlice {label:”C++”;value:30.0}  
PieSlice {label:”Java”;value:45.0}  
PieSlice {label:”C#”;value:30.0}  
}  
}

ChartView {

title:"折线图"

legend.alignment:Qt.AlignTop

antialiasing:true

width:200

height:200

LineSeries {

XYPoint {x:0;y:0}

XYPoint {x:1;y:0.5}

XYPoint {x:2.0;y:0.6}

XYPoint {x:2.3;y:2.0}

XYPoint {x:3.0;y:1.5}

}

}

ChartView {

title:"曲线图"

legend.alignment:Qt.AlignTop

antialiasing:true

width:200

height:200

SplineSeries {

XYPoint {x:0.0;y:0.3}

XYPoint {x:0.3;y:1.3}

XYPoint {x:0.34;y:1.5}

XYPoint {x:0.62;y:4.5}

XYPoint {x:0.7;y:2.1}

}

}

ChartView {

title:"区域图"

width:200

height:200

antialiasing:true

AreaSeries {

upperSeries:LineSeries {

XYPoint {x:0;y:0}

XYPoint {x:1;y:3}

XYPoint {x:4;y:3}

XYPoint {x:5;y:0}

}

}

}

}

Row {

anchors.top:row1.bottom

id:row2

ChartView {

title:"散点图"

width:200

height:200

ValueAxis {

id:valueAxis

min:2007

max:2012

tickCount:6

labelFormat:"%d"

}

ScatterSeries {

axisX:valueAxis

XYPoint {x:2007;y:0.5}

XYPoint {x:2008;y:0.7}

XYPoint {x:2009;y:0.9}

XYPoint {x:2010;y:1.3}

XYPoint {x:2011;y:1.5}

XYPoint {x:2012;y:3.1}

}

}

ChartView {

title:"条状图"

width:200

height:200

BarSeries {

axisX:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

BarSet{label:"中国";values:[2,2,3,4,5,6]}

BarSet{label:"美国";values:[3,4,5,6,7,1]}

BarSet{label:"俄国";values:[2,5,6,7,1,2]}

}

}

ChartView {

title:"层叠条状图"

width:200

height:200

StackedBarSeries {

axisX:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

BarSet {label:"美国";values:[2,2,3,4,5,6]}

BarSet {label:"中国";values:[3,4,5,6,7,1]}

BarSet {label:"日本";values:[4,5,6,1,2,3]}

}

}

ChartView {

title:"百分比条状图"

width:200

height:200

PercentBarSeries {

axisX:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

BarSet {label:"美国";values:[2,3,4,5,6,7]}

BarSet {label:"中国";values:[3,4,5,1,2,3]}

BarSet {label:"日本";values:[4,5,6,1,2,3]}

}

}

}

Row {

anchors.top:row2.bottom

id:row3

ChartView {

title:"水平条状图"

width:200

height:200

antialiasing:true

HorizontalBarSeries {

axisY:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

BarSet {label:"美国";values:[2,3,4,5,6,7]}

BarSet {label:"中国";values:[2,4,2,1,2,4]}

BarSet {label:"俄国";values:[3,4,1,2,4,5]}

}

}

ChartView {

title:"水平层叠条状图"

antialiasing:true

width:200

height:200

HorizontalStackedBarSeries {

axisY:BarCategoryAxis{categories:["2007","2008","2009","2010","2011","2012"]}

BarSet {label:"美国";values:[2,3,2,3,5,6]}

BarSet {label:"中国";values:[2,5,7,1,2,3]}

BarSet {label:"俄国";values:[4,5,1,2,3,1]}

}

}

ChartView {

title:"水平百分比条状图"

antialiasing:true

width:200

height:200

HorizontalPercentBarSeries {

axisY:BarCategoryAxis{categories:["2012","2013","2014","2015","2016"]}

BarSet {label:"美国";values:[2,3,2,1,2,4]}

BarSet {label:"中国";values:[3,4,5,1,2,3]}

BarSet {label:"俄国";values:[2,5,6,1,2,3]}

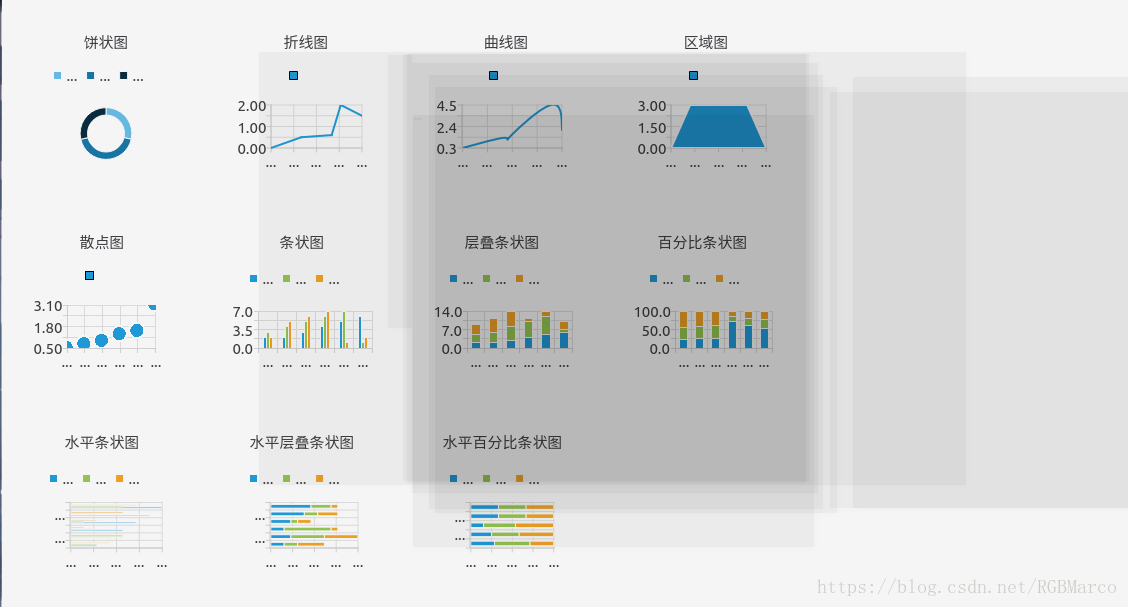
}

}

}

}`

“`



专栏目录

https://kunyu.csdn.net/1.png?p=58&adId=7278&a=7278&c=10229&k=QML%E4%B9%8BQtCharts%E6%A8%A1%E5%9D%97&spm=1001.2101.3001.5002&articleId=80225898&hk=1&d=1&t=3&u=ed0b6ee36fff4a9cbaefd7aa6cd0393c

[qt qml qchart 图表组件](https://download.csdn.net/download/surfsky/8426601" \t "_blank)

02-05

[charts.js 的qml封装，支持各种图表类型：柱形图、饼形图、环形图、折线图、极坐标图、雷达图。自带动画、样式精美，使用方便，代码量小。快抢啊。感觉自己快成王婆了 http://surfsky.cnblogs.com](https://download.csdn.net/download/surfsky/8426601" \t "_blank)

[Qt qml 下使QtCharts\_DreamLife.的博客](https://blog.csdn.net/z609932088/article/details/116455859" \t "_blank)

6-22

[QtQml下使用QtCharts 先说场景,项目中需要用到QtCharts,所以就得搞一下了,最简单的方式还是直接看Qt的帮助文档。 1.1 引入文档 根据帮助文档显示,我们只需要引入对应的包就可以了,在项目中引入QtCharts模块,如下 ...](https://blog.csdn.net/z609932088/article/details/116455859" \t "_blank)

[qchart 图表\_Qt开发技术:QtCharts(一)QtCharts基本介绍...](https://blog.csdn.net/weixin_39866419/article/details/111514932" \t "_blank)

4-27

[Qt自带的二维图标QCharts相关研发笔记。 Qt图标(Qt Charts) 概述 Qt图表模块提供了一组易于使用的图表组件。它使用Qt图形视图框架,因此图表可以很容易地集成到现代用户界面中。Qt图表可以用作QWidgets、QGraphicsWidget或QML类型。用户可以...](https://blog.csdn.net/weixin_39866419/article/details/111514932" \t "_blank)

[QT: QML使用QtCharts模块](https://blog.csdn.net/yhjahjj1314/article/details/88682511" \t "_blank)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 1175

[首先，在pro中要添加charts,widgets，即 　　　　Qt += qml quick widgets charts 在main.cpp中，添加头文件 #include <QApplication>( 注， pro中要先添加widgets，不然该头文件下会显示横线，表明找不到），并在main 函数修改为下面的形式： 1 2 3 4 ...](https://blog.csdn.net/yhjahjj1314/article/details/88682511" \t "_blank)

[Qt qml 下使QtCharts](https://dreamlife.blog.csdn.net/article/details/116455859" \t "_blank)

[DreamLife](https://blog.csdn.net/z609932088)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 433

[欢迎来到我的博客，希望这篇文章对你有所帮助，如果觉得不错，请点赞搜藏哈。 文章目录Qt Qml 下使用QtCharts1.1 引入文档1.2 在pro中包含 charts1.3 运行官方例子1.4 修改main文件1.5 手动拷贝 相关文件 Qt Qml 下使用QtCharts 先说场景，项目中需要用到QtCharts，所以就得搞一下了，最简单的方式还是直接看Qt的帮助文档。 1.1 引入文档 根据帮助文档显示，我们只需要引入对应的包就可以了，在项目中引入QtCharts模块，如下 import QtCh.](https://dreamlife.blog.csdn.net/article/details/116455859" \t "_blank)

[Qt开发总结(19)——Qt Charts\_bjtuwayne的博客\_qtcharts](https://blog.csdn.net/bjtuwayne/article/details/103642828" \t "_blank)

7-15

[Qt Charts模块提供以下几类图表类型,每个图表都通过QAbstractSeries来实现具体形式。具体来说就是把series类的对象添加到QChart or ChartView对象中。当然,你可以在一个图表中混合添加多种曲线。](https://blog.csdn.net/bjtuwayne/article/details/103642828" \t "_blank)

[QML使用QtCharts 报错\_FlyWM\_的博客](https://blog.csdn.net/a844651990/article/details/78972071" \t "_blank)

6-6

[意思大概就是要在qtquick中使用QtCharts模块的话必须使用QApplication代替QGuiApplication。 pro文件中添加: QT += widgets 1 修改main.cpp为: #include <QApplication>#include <QQmlApplicationEngine>intmain(intargc,char\*argv[]) ...](https://blog.csdn.net/a844651990/article/details/78972071" \t "_blank)

[qml: QtCharts模块的使用（基本配置)------<一>](https://blog.csdn.net/weixin_30699955/article/details/96978817" \t "_blank)

[weixin\_30699955的博客](https://blog.csdn.net/weixin_30699955)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 218

[QtCharts模块可以用于绘制图表； 导入模块： import QtCharts 2.2例子： import QtQuick 2.0 import QtCharts 2.2 ChartView { width: 400 height: 300 theme: ChartView.ChartThemeBrownSand antialiasing:...](https://blog.csdn.net/weixin_30699955/article/details/96978817" \t "_blank)

[Qt Charts QML下的图形绘制方法](https://blog.csdn.net/weixin_41922484/article/details/123045310" \t "_blank)

[weixin\_41922484的博客](https://blog.csdn.net/weixin_41922484)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 1284

[目录一、参考二、简介三、静态绘制曲线图四、动态绘制曲线图和散点图五、如何确定哪些QML对象属性可以被JS调用六、如何确定某些QML属性赋值的格式 一、参考 https://doc.qt.io/qt-5/qtcharts-index.html 二、简介 Qt Charts 提供了一系列使用图表功能的简单方法。它使用Qt Graphics View Framework 图形视图框架，因此可以很容易集成到用户界面。可以使用Qt Charts作为QWidgets, QGraphicsWidget, 或者 Q.](https://blog.csdn.net/weixin_41922484/article/details/123045310" \t "_blank)

[麒麟系统Qt5.6.1中加装QtChart模块\_鱼月半的博客](https://blog.csdn.net/baidu_31788709/article/details/116239285" \t "_blank)

6-22

[上述过程不要指定地址了,就让他默认地址放置即可,以免Qt找不到他了。 这时候,主要有这两个变化 /usr/lib/aarch64-linux-gnu/qt5/qml/QtCharts //这个目录下会生成QtCharts /usr/include/aarch64-linux-gnu/qt5/QtCharts //这...](https://blog.csdn.net/baidu_31788709/article/details/116239285" \t "_blank)

[qml庞大数据量绘图QCharts性能提升](https://blog.csdn.net/qq_47898198/article/details/119864372" \t "_blank)

[qq\_47898198的博客](https://blog.csdn.net/qq_47898198)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 525

[qml庞大数据量绘图性能提升问题以及解决思路实验方案方案一：在cpp里绘制好要画的线或者波形图再传到qml里面展示方案二：数据分组在不同信号里，在不同事件中对一张图进行绘制方案三：并行异步，创建新的线程进行大数据量图的绘制方案四：对处理好的数据先存在临时文件里，之后以传指针给qml，滑动赋值方案五：在风险承受范围内每隔n个点绘制一个点（有风险）总结注意 问题以及解决思路  qml在绘制20M起步的大规模数据的图标如波形图频谱等时会遇到绘图效率极其低下且有心者会发现在逐个加点绘图的过程中，速度效率会越来越慢，](https://blog.csdn.net/qq_47898198/article/details/119864372" \t "_blank)

[QML中QCharts饼图自定义（Doughnut和Nightingale）](https://gongjianbo1992.blog.csdn.net/article/details/90143097" \t "_blank)

[龚建波](https://blog.csdn.net/gongjianbo1992)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 3968

[0.前言 虽然js的ECharts很好看，但是得嵌一个webengineview，如果是Loader加载的话会很慢，所以我试着用QCharts来自定义一些简单的图表。 UI原型图如下： 最终实现效果如下： 相对来说，没有ECharts的好看，花样也少，而且要定制效果更好的话得参考源码从C++入手改。比较坑的是hover只有true和false两个状态，没有鼠标坐标，没法做标...](https://gongjianbo1992.blog.csdn.net/article/details/90143097" \t "_blank)

[Qt Chart在Qml中初体验](https://blog.csdn.net/LHRui_1/article/details/83834753" \t "_blank)

[LHRui\_1的博客](https://blog.csdn.net/LHRui_1)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 2816

[Chart之初体验Qml与QCustomPlot准备使用实时曲线Qml中获取数据ChartView的Scroll函数让曲线移动起来坐标轴Range动起来坐标动起来，然后replaceQml的坑 Qml与QCustomPlot 最近想在Qml中实现绘制实时曲线，一开始想到了好用且强大的[\*QCustomPlot][1]，但伤心的发现其在Qml中无法使用。因为其是基于QWidget开发而来的第三方库，...](https://blog.csdn.net/LHRui_1/article/details/83834753" \t "_blank)

[QCharts 在Qt及QML上的应用 使用QtCharts 报错](https://blog.csdn.net/qq_24890953/article/details/99305847" \t "_blank)

[你学习了吗？咋办，先想法子忽悠成“砖家”呗](https://blog.csdn.net/qq_24890953)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 443

[Chart 上的点显示文字 QML 必须满足 1 .pro 文件添加 QT += charts 2 使用 QApplication 来代替默认的 QGuiApplication 3 软件必须安装了 QtCharts 模块（默认不安装需要自己选上） #include <QQmlApplicationEngine> #include <QApplication>...](https://blog.csdn.net/qq_24890953/article/details/99305847" \t "_blank)

[QML Chart之PieSeries系列(饼图)](https://blog.csdn.net/qq1071247042/article/details/53048815" \t "_blank)

[我愿用一生守护你的笑容](https://blog.csdn.net/qq1071247042)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 2680

[QML ChartView PieSeries(图表之饼图系列)](https://blog.csdn.net/qq1071247042/article/details/53048815" \t "_blank)

[Qt图表绘制（QtCharts）-QtCharts基本介绍及包含的C++类（1）](https://blog.csdn.net/qq_43627907/article/details/124389945" \t "_blank)

[最新发布](https://blog.csdn.net/qq_43627907/article/details/124389945" \t "_blank)

[mahuifa的博客](https://blog.csdn.net/qq_43627907)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 1977

[Qt Charts 模块提供了一组易于使用的图表组件，能够创建时尚、交互式、以数据为中心的用户界面。 它使用 `Qt Graphics View `框架，因此可以轻松地将图表集成到现代用户界面中。 Qt Charts 可以用作 QWidgets、QGraphicsWidget 或 QML 类型。 用户可以通过选择图表主题之一轻松创建令人印象深刻的图表。](https://blog.csdn.net/qq_43627907/article/details/124389945" \t "_blank)

[qt label显示图片\_Qt官方示例-QML Axes](https://blog.csdn.net/weixin_39592240/article/details/110980282" \t "_blank)

[weixin\_39592240的博客](https://blog.csdn.net/weixin_39592240)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 170

[QML轴线图示例，折线图，散点图。使用相同轴坐标的折线图和散点图。代码：ChartView { title: "Two Series, Common Axes" anchors.fill: parent legend.visible: false antialiasing: true ValueAxis { id: axisX ...](https://blog.csdn.net/weixin_39592240/article/details/110980282" \t "_blank)

[【Qt6】【QML】【教程】QML动态柱状图](https://blog.csdn.net/qq_38844263/article/details/122618970" \t "_blank)

[qq\_38844263的博客](https://blog.csdn.net/qq_38844263)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 428

[上节我们解决了高分辨鼠标坐标偏移的问题，这次我们来整个动态柱状图来实现实时音频频谱显示 完整项目MediaState软件项目地址：https://gitee.com/txwh-media-state 教程针对MediaStateT 先来看效果： 效果感觉还行？将就吧，到时候抽时间再美化美化。。而且似乎得上个多线程。。 来讲实现： 这里一共两个声道，我们就讲一个声道，剩下那个一样的实现。先创建一个基本构架。 ChartView{ id:left\_bar // t](https://blog.csdn.net/qq_38844263/article/details/122618970" \t "_blank)

[qt中qtchart图表模块47个例程预览](https://rong11417.blog.csdn.net/article/details/100778648" \t "_blank)

[rong11417的博客](https://blog.csdn.net/rong11417)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 1530

[qtchart总结 linux中chart类在 examples例子的目录是在/usr/lib64/qt5/examples/charts中 1.barchart 2.barmodelmapper 3.boxplotchart 4.callout 5.candlestickchart 6.chartinteractions 7.chartthemes...](https://rong11417.blog.csdn.net/article/details/100778648" \t "_blank)

[qml +echart +WebEngineView+js 构建实时图表](https://blog.csdn.net/lantian510/article/details/122820963" \t "_blank)

[墨鸦](https://blog.csdn.net/lantian510)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 841

[今天正月初八，马上要上班了哈。 之前我写过的一篇文章，关于qml里面chart来绘制实时曲线的，后来发现qml的chart的组件存在一些问题，特别是数据一多，就容易卡，绘制界面不流畅等问题 qml Qchart 时间轴的图表\_墨鸦-CSDN博客\_qml时间轴 对于一个晚期强迫症患者来说，必须要找到一种 js的解决办法，因为C++把数据发给了前端页面qml，如果qml还要回去调用C++的一些类啊，方法啊，这种做法显然不够优雅，但是qml自带的chart组件绘制静态图表可以，但是对于绘制实时图表，还是.](https://blog.csdn.net/lantian510/article/details/122820963" \t "_blank)

[QML在Charts中动态修改Series（LineSeries, ScatterSeries, ，SplineSeries）的值](https://blog.csdn.net/zjgo007/article/details/120277206" \t "_blank)

[zjgo007的专栏](https://blog.csdn.net/zjgo007)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 592

[项目需要对Charts中的图表动态的修改Series中数据的值，通过查看帮助文档，发现QML Charts中关于XYSeries的方法的说明如下图，共7个方法，其中数据替换的方法replace只写了一种通过点的值进行替换的方法。](https://blog.csdn.net/zjgo007/article/details/120277206" \t "_blank)

[qml 导航栏TabBar 工具栏ToolBar](https://blog.csdn.net/qq_33373173/article/details/109954246" \t "_blank)

[qq\_33373173的博客](https://blog.csdn.net/qq_33373173)

https://csdnimg.cn/release/blogv2/dist/pc/img/readCountBlack.png 2334

[导航栏 TabBar 提供基于选项卡的导航模型，允许用户在不同的视图或子任务之间切换。用TabButton控件填充，可以理解为button按钮控件。一般会与布局或容器控件一起使用 属性 contentHeight : 保存内容的高度。它用于计算导航栏的总隐式高度 contentWidth : 保存内容的宽度。它用于计算导航栏的总隐式宽度 position : 保存导航栏的位置 附加属性 TabBar.index：[只读]，保存TabBar中每个导航栏按钮的索引 TabBar.position：[只读]，保](https://blog.csdn.net/qq_33373173/article/details/109954246" \t "_blank)