

Контекстно-независимое представление растровых изображений

QImage **GIF BMP JPG XMP XBM**
e **PNG**
scanLine()

QImage img(320, 240, QImage::Format_RGB32) //32- глубина цвета.

QImage img (“D:\\1.jpg”);

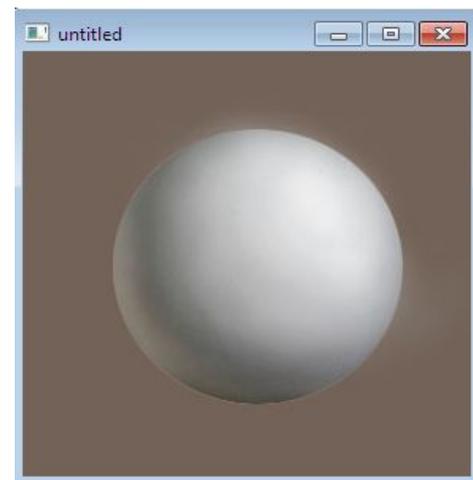
QImage img;
img.load(“D:\\1.jpg”);

img.save(“1.jpg”,”JPG”);



Контекстно-независимое представление растровых изображений

```
1  #include "mainwindow.h"
2  #include <QApplication>
3  #include <QPainter>
4  #include <QDebug>
5
6
7  class MyWindow: public QMainWindow{
8  protected:
9      virtual void paintEvent(QPaintEvent * e){
10         QMainWindow::paintEvent(e);
11         QPainter painter(this);
12         QImage img("E:\\1.jpg");
13         painter.drawImage(0,0,img);
14     }
15 };
16
17
18 int main(int argc, char *argv[])
19 {
20     QApplication a(argc, argv);
21     MyWindow w;
22     QImage img("E:\\1.jpg");
23     w.setGeometry(50,50,img.width(),img.height());
24     w.show();
25     return a.exec();
26 }
```



**QPainter::drawImage(
)**

Контекстно-независимое представление растровых изображений

`fill()`;

```
QImage img("E:\\1.jpg");  
QRgb value;  
value = qRgb(122, 163, 39);  
img.fill(value);  
painter.drawImage(0,0,img);
```



Контекстно-независимое представление растровых изображений

pixel(x,y)

```
QRgb rgb = img.pixel(250,100);
```

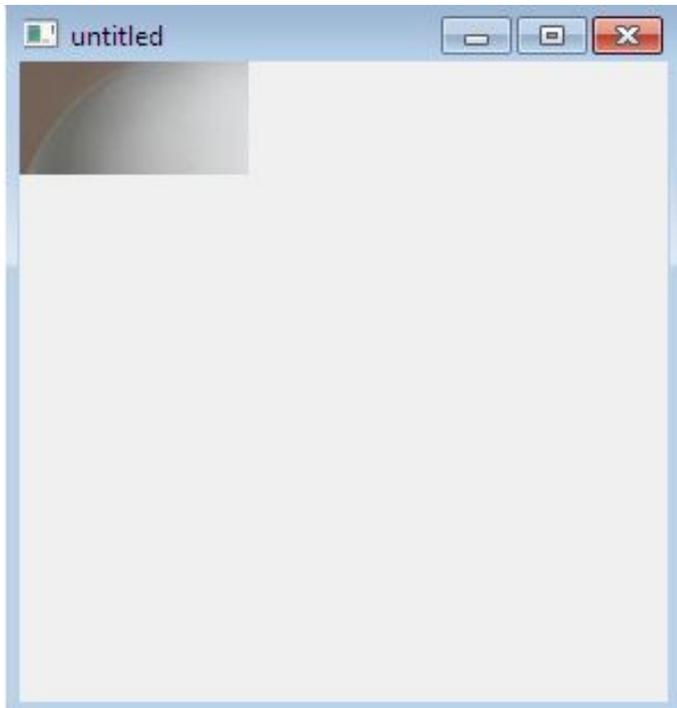
setPixel(x,y,rgb);

```
QRgb rgb = qRgb(200,100,0);  
img.setPixel(20,50,rgb);
```



Контекстно-независимое представление растровых изображений

```
painter.drawImage(0, 0, img, 60, 60, 100, 50);
```



invertPixels()

scaled()

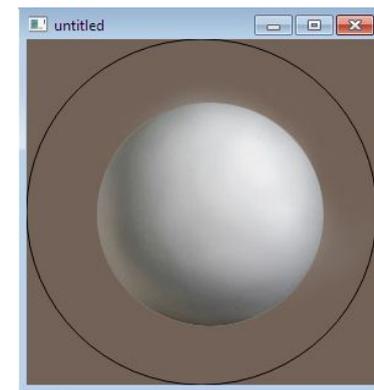
mirrored()



Контекстно-независимое представление растровых изображений

```
class MyWindow: public QMainWindow{
protected:
    virtual void paintEvent(QPaintEvent * e){
        QMainWindow::paintEvent(e);
        QImage img("E:\\1.jpg");
        QPainter painter;
        painter.begin(&img);
        painter.initFrom(this);
        painter.setRenderHint(QPainter::Antialiasing, true);
        painter.drawEllipse(0,0,img.width(),img.height());
        painter.end();

        painter.begin(this);
        painter.drawImage(0,0,img);
        painter.end();
    }
};
```



Контекстно-зависимое представление растровых изображений

QPixmap

```
QPixmap pix(300,300);
```

```
QPixmap::defaultDepth()
```

```
QPixmap pix("E:\\1.jpg");
```

```
load()    save()
```



Контекстно-зависимое представление растровых изображений

**QPainter::drawPixmap(
)**

```
QPainter painter(this);  
QPixmap pix("D:\\q\\pict\\untitled\\1.jpg");  
painter.drawPixmap(0,0,pix);
```

```
QPainter painter(this);  
QPixmap pix("D:\\q\\pict\\untitled\\1.jpg");  
QRect r(0,0, pix.width(), pix.height()/3);  
painter.drawPixmap(r,pix);
```



Контекстно-зависимое представление растровых изображений

QPixmapCache

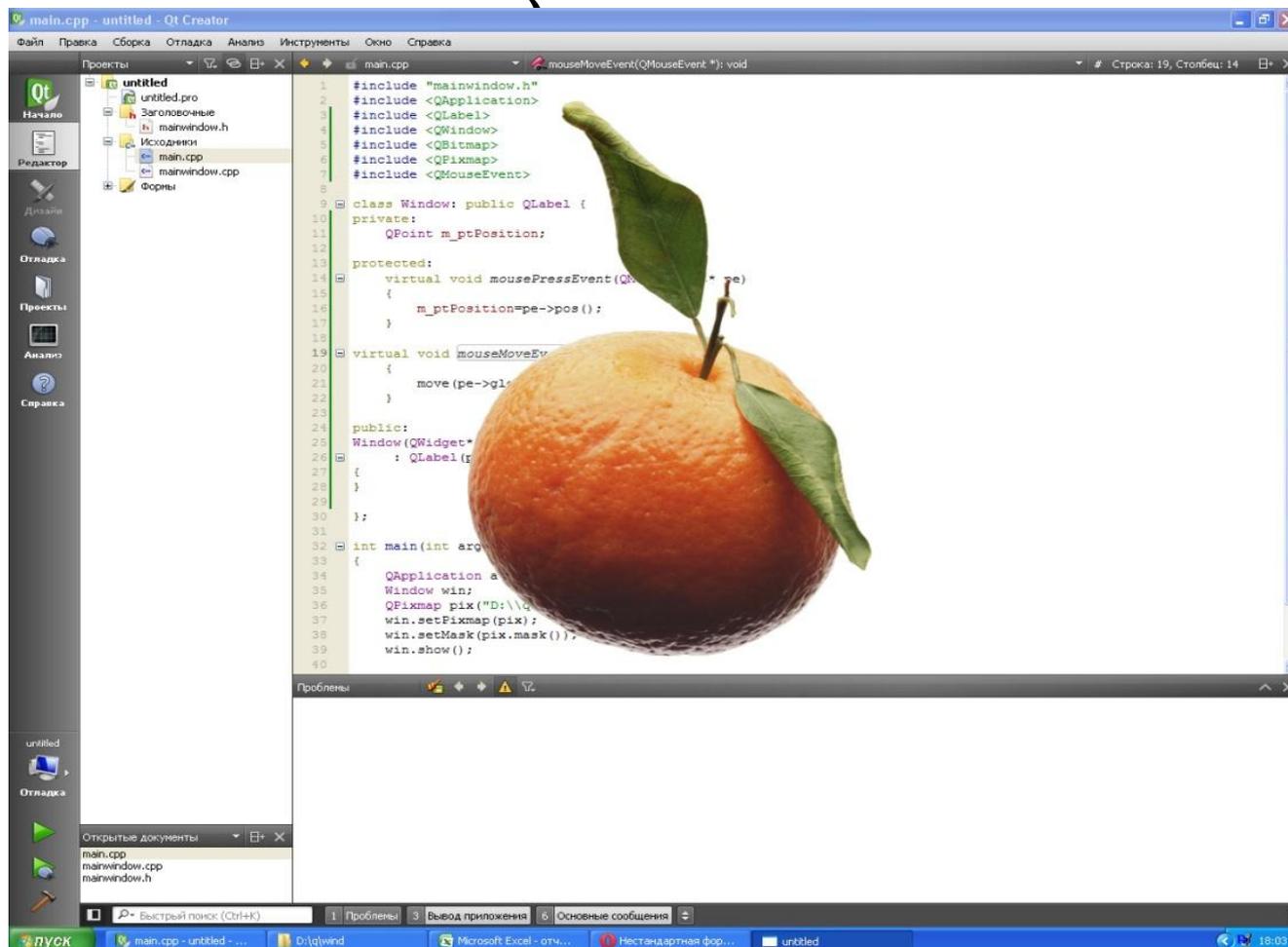
insert()

find()



Контекстно-зависимое представление растровых изображений

setMask(



Контекстно-зависимое представление растровых изображений

```
30
31 int main(int argc, char *argv[])
32 {
33     QApplication a(argc, argv);
34     Window win;
35     QPixmap pix("D:\\q\\wind\\3.png");
36     win.setPixmap(pix);
37     win.setMask(pix.mask());
38     win.show();
39
40     return a.exec();
```



```
#include "mainwindow.h"
#include <QApplication>
#include <QLabel>
#include <QWindow>
#include <QBitmap>
#include <QPixmap>
#include <QMouseEvent>

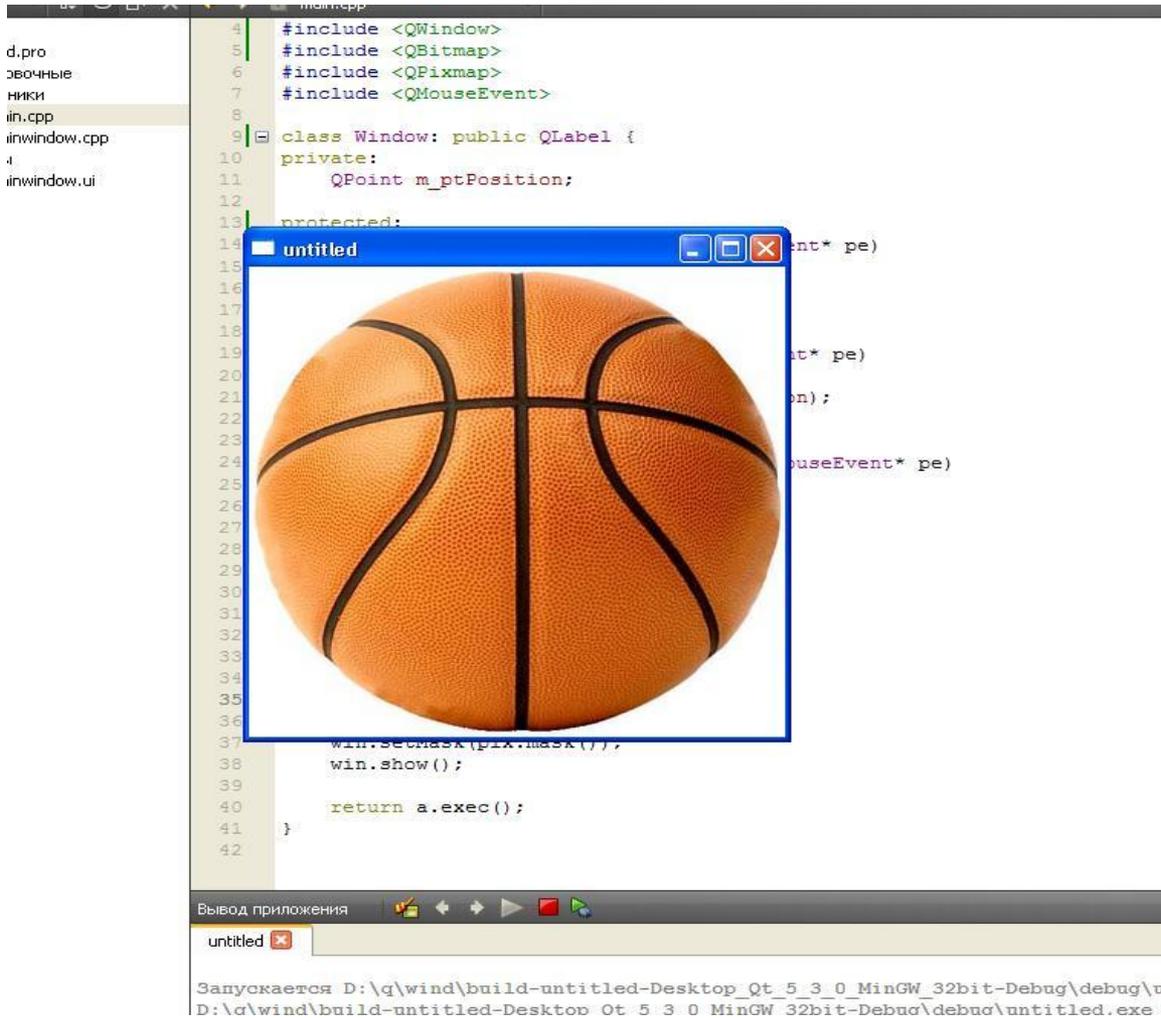
class Window: public QLabel {
private:
    QPoint m_ptPosition;

protected:
    virtual void mousePressEvent(QMouseEvent* pe)
    {
        m_ptPosition=pe->pos();
    }

    virtual void mouseMoveEvent(QMouseEvent* pe)
    {
        move(pe->globalPos() - m_ptPosition);
    }

    virtual void mouseDoubleClickEvent(QMouseEvent* pe)
    {
        exit(0);
    }
};
```

Контекстно-зависимое представление растровых изображений



The screenshot displays a Qt IDE environment. On the left, a file explorer shows a project structure with files like `d.pro`, `mainwindow.cpp`, and `mainwindow.ui`. The main editor window shows a C++ source file with the following code:

```
4 #include <QWindow>
5 #include <QBitmap>
6 #include <QPixmap>
7 #include <QMouseEvent>
8
9 class Window: public QLabel {
10 private:
11     QPoint m_ptPosition;
12
13 protected:
14
15     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt) : QLabel(parent, pix, pt) {}
16
17     Window(QWidget* parent, const QPixmap& pix) : QLabel(parent, pix) {}
18
19     Window(QWidget* parent) : QLabel(parent) {}
20
21     Window() {}
22
23     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text) : QLabel(parent, pix, pt, text) {}
24
25     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color) : QLabel(parent, pix, pt, text, color) {}
26
27     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font) : QLabel(parent, pix, pt, text, color, font) {}
28
29     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name) : QLabel(parent, pix, pt, text, color, font, name) {}
30
31     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2) : QLabel(parent, pix, pt, text, color, font, name, name2) {}
32
33     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2, const QString& name3) : QLabel(parent, pix, pt, text, color, font, name, name2, name3) {}
34
35     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2, const QString& name3, const QString& name4) : QLabel(parent, pix, pt, text, color, font, name, name2, name3, name4) {}
36
37     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2, const QString& name3, const QString& name4, const QString& name5) : QLabel(parent, pix, pt, text, color, font, name, name2, name3, name4, name5) {}
38
39     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2, const QString& name3, const QString& name4, const QString& name5, const QString& name6) : QLabel(parent, pix, pt, text, color, font, name, name2, name3, name4, name5, name6) {}
40
41     Window(QWidget* parent, const QPixmap& pix, const QPoint& pt, const QString& text, const QColor& color, const QFont& font, const QString& name, const QString& name2, const QString& name3, const QString& name4, const QString& name5, const QString& name6, const QString& name7) : QLabel(parent, pix, pt, text, color, font, name, name2, name3, name4, name5, name6, name7) {}
42 }
```

The code defines a `Window` class that inherits from `QLabel`. It includes headers for `QWindow`, `QBitmap`, `QPixmap`, and `QMouseEvent`. The class has a private member `m_ptPosition` and a protected constructor that takes a `QWidget*`, a `QPixmap`, and a `QPoint`. The `main` function creates a `Window` object, sets its mask, and shows it.

In the center, a window titled "untitled" displays a high-resolution image of a basketball. The image is centered and occupies most of the window's area.

At the bottom, the "Вывод приложения" (Application Output) window shows the command used to run the application:

```
Запускается D:\q\wind\build-untitled-Desktop_Qt_5_3_0_MinGW_32bit-Debug\debug\untitled.exe s
```

Контекстно-зависимое представление растровых изображений

public:

```
Window(QWidget* pwgt = 0)  
    :QLabel(pwgt, Qt::FramelessWindowHint |  
Qt::Window)  
    {}
```



Использование каскадных стилей документа

CSS *.qss

QApplication::setStyleSheet()

QWidget::setStyleSheet()

a.setStyleSheet("описание стиля")

```
1 #include "mainwindow.h"
2 #include <QApplication>
3 #include <QFile>
4 #include <QStyle>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     MainWindow w;
10
11     QFile file("D:\\q\\styl\\s1.qss");
12     file.open(QFile::ReadOnly);
13     QString strCSS = QLatin1String(file.readAll());
14     a.setStyleSheet(strCSS);
15     w.show();
16     return a.exec();
17 }
18
```



Использование каскадных стилей документа

селектор {свойство: значение}

```
QPushButton {color: blue}
```

```
QLabel {  
    color: black;  
    background-color: red;  
}
```

```
QLabel {  
    color: rgb(255,0,0);  
    background-color: #FFFFFF;  
}
```



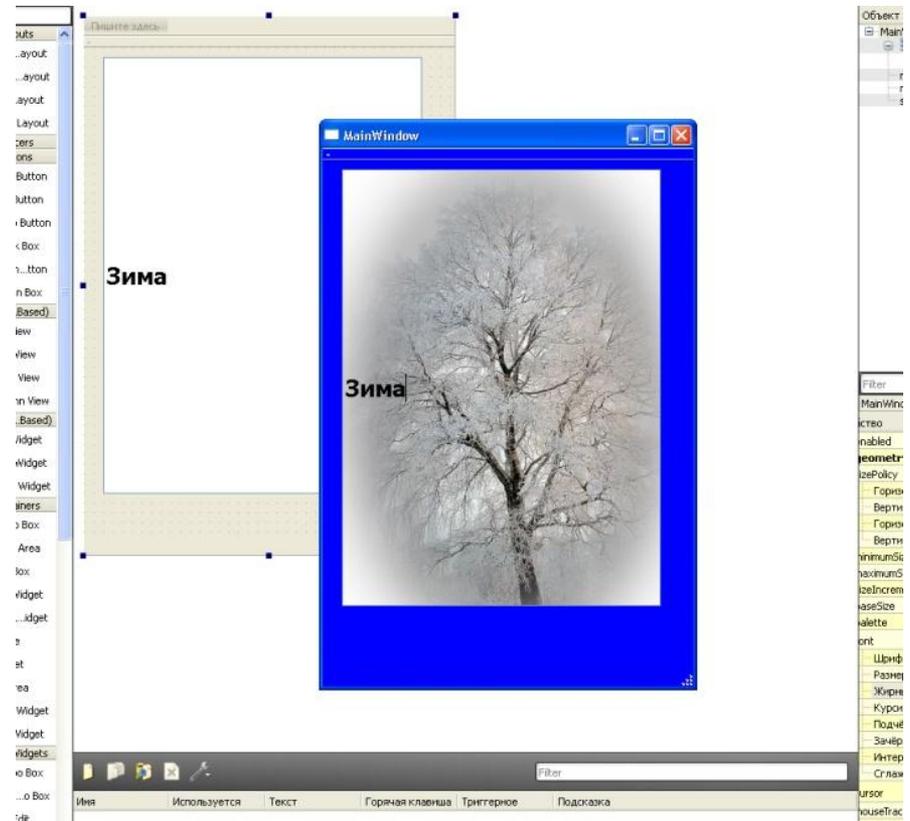
Использование каскадных стилей документа

```
QPushButton {  
  color: green;  
  border: 1px solid black;  
  border-radius: 5px;  
  background: qlineargradient(x1:0, y1:1, x2:0, y2: 0,  
                              stop:1 rgb(133,133,135),  
                              stop:0.4 rgb(31,31,33));  
}
```



Использование каскадных стилей документа

```
QLineEdit {  
background-image: url (D:/q/1.png) ;  
}  
QMainWindow {  
background-color: blue ;  
}
```



Использование каскадных стилей документа

```
QLineEdit, QLabel, QPushButton {color: red}
```

```
.PushButton {color: red}
```

```
QLabel#MyLabel
```



Использование каскадных стилей документа

```
Namespace My {  
Class My : public QWidget {....}; }
```

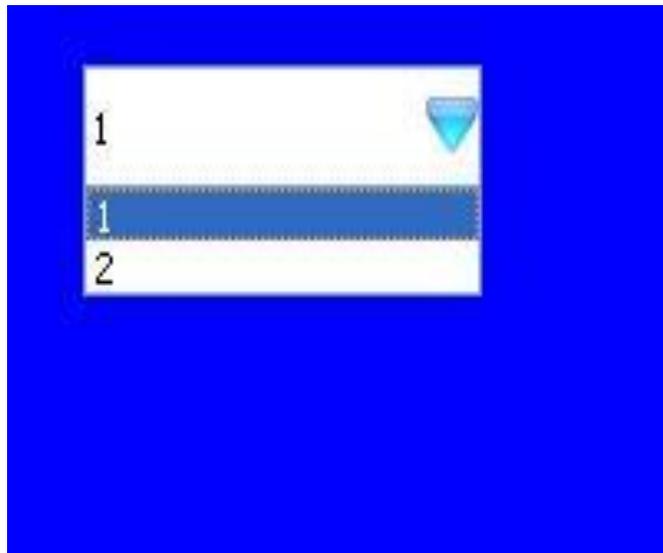
```
/* .qss */  
My {  
    color: red;  
    background-color: blue  
}
```

```
/* КОММЕНТ */
```



Использование каскадных стилей документа

```
QComboBox::drop-down {image: url(D:/q/styl/3.png) }
```



Использование каскадных стилей документа

Подэлемент	Описание	Возможные виды
::down-arrow	Стрелка вниз. Имеется, например, у виджета выпадающего списка и у счетчика	
::down-button	Кнопка вниз. Имеется у виджета счетчика	
::drop-down	Стрелка виджета выпадающего списка	
::indicator	Индикатор кнопки флажка или переключателя, а также группировки кнопок	
::item	Элемент меню, строки состояния	
::menu-indicator	Индикатор меню кнопки нажатия, обычно это стрелка	

Подэлемент	Описание	Возможные виды
::title	Надпись группы	Title
::up-arrow	Стрелка вверх. Имеется у виджета счетчика	
::up-button	Кнопка вверх. Имеется у виджета счетчика	

Использование каскадных стилей документа

```
QPushButton: hover {color: red}
```

```
QLineEdit: hover {color: red}
```

Обозначение	Описание
:checked	Активировано
:closed	Виджет находится в закрытом либо свернутом состоянии
:disabled	Виджет недоступен
:enabled	Виджет доступен
:focus	Виджет находится в фокусе ввода
:hover	Указатель мыши находится над виджетом
:indeterminate	Кнопка находится в промежуточном неопределенном состоянии
:off	Выключено (для виджетов, которые могут быть в фиксированном состоянии нажато/не нажато)
:on	Включено (для виджетов, которые могут быть в фиксированном состоянии нажато/не нажато)
:open	Виджет находится в открытом или развернутом состоянии
:pressed	Виджет был нажат мышью
:unchecked	Деактивировано

Использование каскадных стилей документа

```
QCheckBox:hover:checked {color: red}
```

```
QCheckBox:hover, QCheckBox:checked {color: red}
```

```
QLineEdit:!hover {color: red}
```



Использование каскадных стилей документа

```
QPushButton:hover {  
background: qlineargradient(x1:0, y1:1, x2:0, y2:0,  
                             stop:1 rgb(133,133,135),  
                             stop:0.4 rgb(31,31,33),  
                             stop:0.2 rgb(0,0,150));  
}
```



Использование каскадных стилей документа

```
QPushButton:pressed {  
background: qlineargradient(x1:0, y1:1, x2:0, y2:0,  
                             stop:0 rgba(1,1,5,80),  
                             stop:0.6 rgba(18,18,212,80),  
                             stop:0.5 rgba(142,142,245,80));  
}
```

