

# Объектно-ориентированное программирование

## Лекция 12. Qt

# Установка

**Шаг 1.** Скачать установщик Qt 5

[Официальный online-установщик](#)

[Неофициальный источник 1](#)

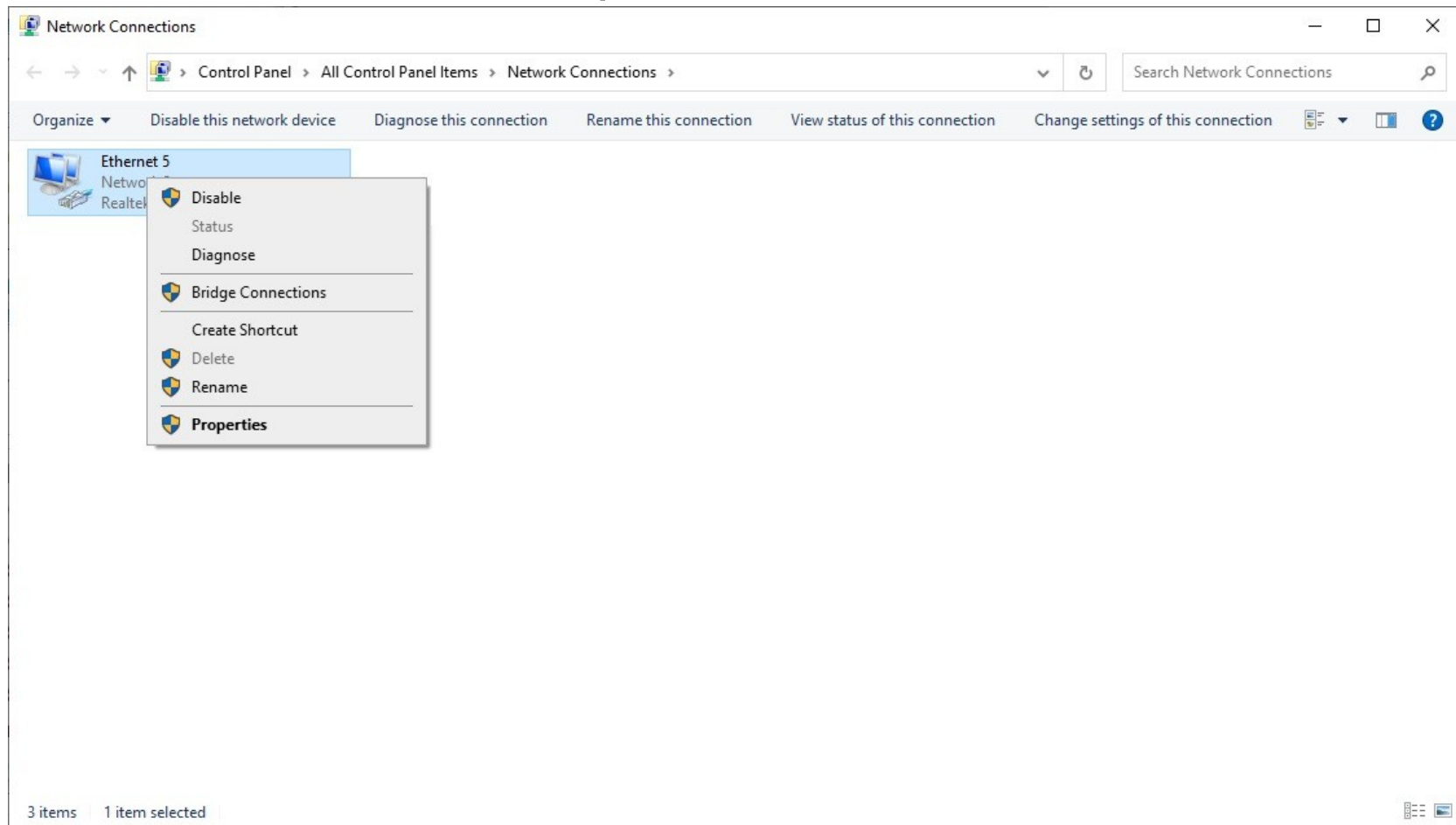
[Неофициальный источник 2](#)

**Шаг 2.** Отключить интернет (см. слайд 3)

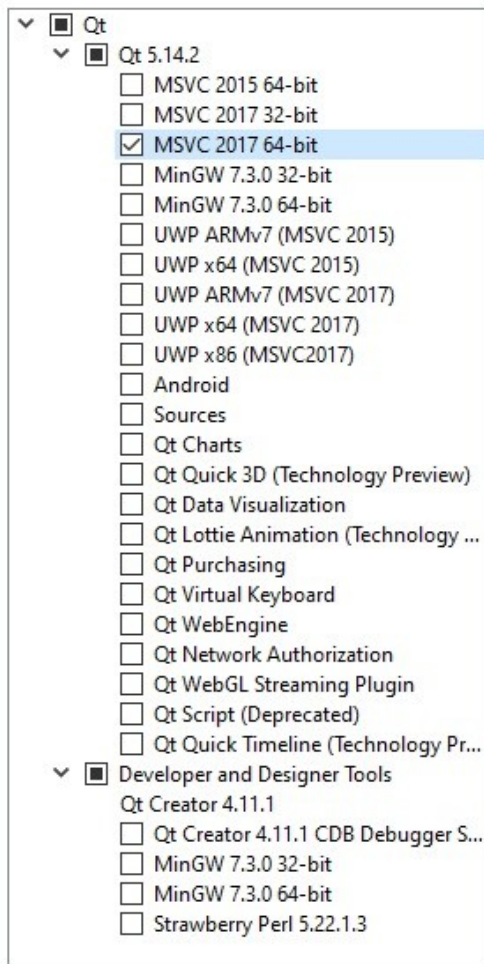
**Шаг 3.** Запустить от имени администратора

**Шаг 4.** Установить компонент «MSVC 2017 64-bit» (см. слайд 4)

# Установка: отключение интернета



# Установка: устанавливаемые компоненты



# Настройка проекта

**Шаг 1.** Создать/открыть проект CMake в VS

**Шаг 2.** В меню «Project» выбрать пункт «CMake Settings».

В поле «CMake command arguments» добавить путь к установленной библиотеке:

-DCMAKE\_PREFIX\_PATH="C:\Qt\5.14.2\5.14.2\msvc2017\_64\lib\cmake"

**Шаг 3.** В файле CMakeLists.txt (см. слайд 6):

- 1) добавить зависимость от Qt;
- 2) включить [moc](#);
- 3) включить копирование динамических библиотек Qt в выходную директорию сборки;
- 4) опционально изменить тип проекта для отключения консоли.

# Настройка проекта: CMakeLists.txt

```
find_package(Qt5 COMPONENTS Core Gui Widgets)

set(CMAKE_AUTOMOC ON)

add_executable(${PROJECT_NAME} ...)
target_link_libraries(${PROJECT_NAME} Qt5::Core Qt5::Gui Qt5::Widgets)

if (WIN32)
    # Копирование DLL-файлов библиотеки Qt в выходную директорию
    add_custom_command(
        TARGET ${PROJECT_NAME} POST_BUILD
        COMMAND ${Qt5_DIR}/../../../../bin/windeployqt $<TARGET_FILE:${PROJECT_NAME}>
    )

    # Чтобы убрать консоль
    set_property(TARGET ${PROJECT_NAME} PROPERTY WIN32_EXECUTABLE TRUE)
endif (WIN32)
```

# Настройка проекта: проверка работоспособности

```
#include <QApplication>
#include <QWidget>

int main(int argc, char* argv[]) {
    QApplication app(argc, argv);

    QWidget widget;
    widget.show();

    return QApplication::exec();
}
```

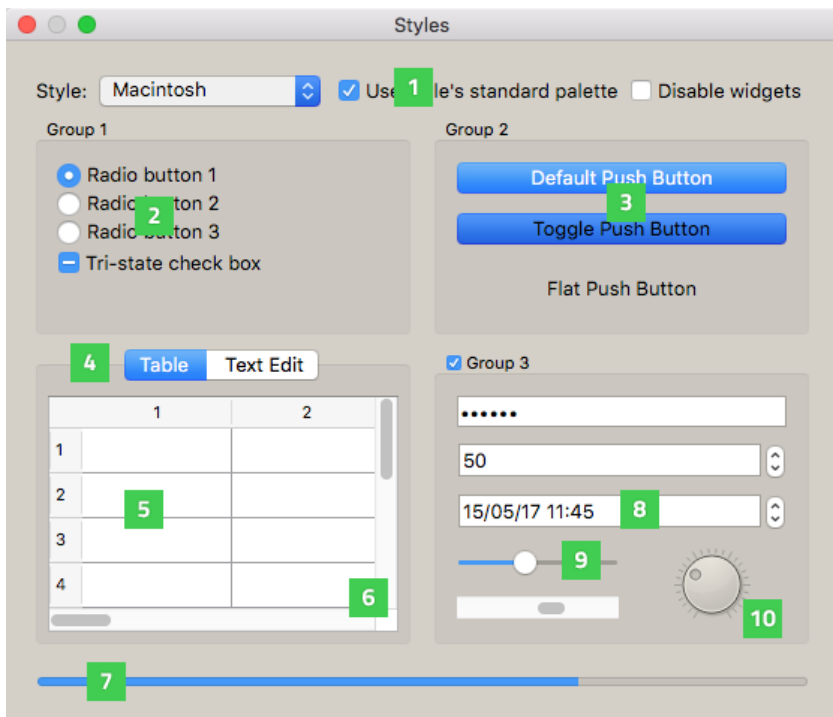
# Документация

 [Документация](#)

[Конструктор QApplication](#) (аргументы командной строки)



# Элементы управления



[QCheckBox](#) (1) provides a checkbox with a text label.

[QRadioButton](#) (2) provides a radio button with a text or pixmap label.

[QPushButton](#) (3) provides a command button.

[QTabWidget](#) (4) provides a stack of tabbed widgets.

[QTableWidget](#) (5) provides a classic item-based table view.

[QScrollBar](#) (6) provides a vertical or horizontal scroll bar.

[QProgressBar](#) (7) provides a horizontal progress bar.

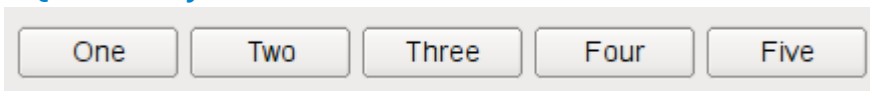
[QDateTimeEdit](#) (8) provides a widget for editing dates and times.

[QSlider](#) (9) provides a vertical or horizontal slider.

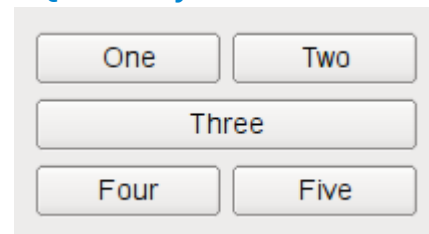
[QDial](#) (10) provides a rounded range control.

# Диспетчеры компоновки

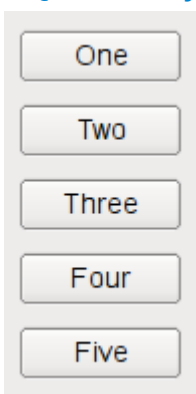
[QHBoxLayout](#)



[QGridLayout](#)



[QVBoxLayout](#)



[QFormLayout](#)



# Диспетчеры компоновки

First operand:

Second operand:

Result:

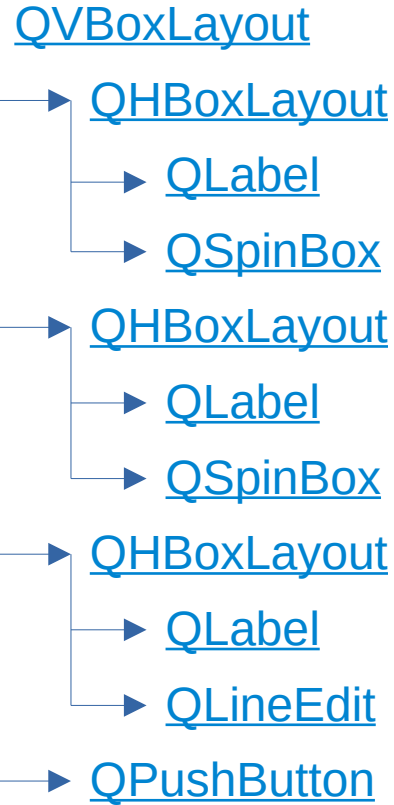
# Диспетчеры компоновки

First operand: 2

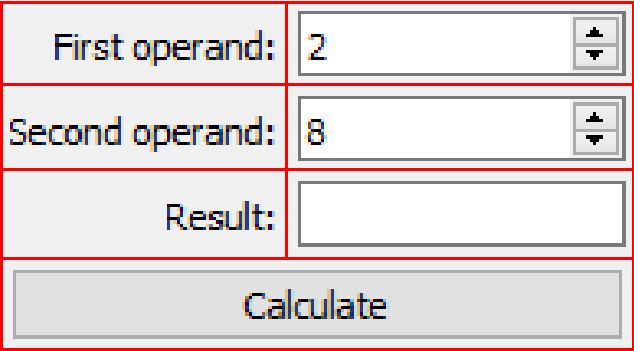
Second operand: 8

Result:

Calculate



# Диспетчеры компоновки



First operand:	2
Second operand:	8
Result:	
Calculate	

## QGridLayout

- ▶ QLabel
- ▶ QSpinBox
- ▶ QLabel
- ▶ QSpinBox
- ▶ QLabel
- ▶ QLineEdit
- ▶ QPushButton

*row=0, column=0*

*row=0, column=1*

*row=1, column=0*

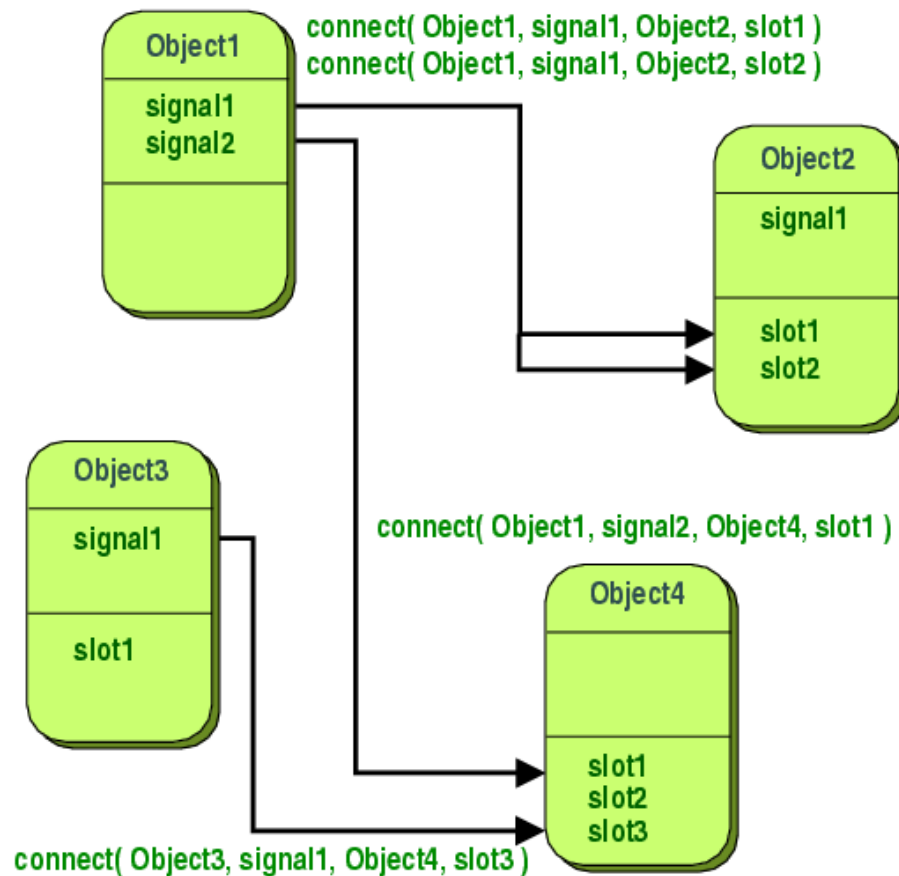
*row=1, column=1*

*row=2, column=0*

*row=2, column=1*

*row=3, column=1, columnSpan=2*

# Сигналы и слоты



# Сигналы и слоты

## QAbstractSlider Class

The QAbstractSlider class provides an integer value within a range. [More...](#)

## Signals

void	<code>actionTriggered(int action)</code>
void	<code>rangeChanged(int min, int max)</code>
void	<code>sliderMoved(int value)</code>
void	<code>sliderPressed()</code>
void	<code>sliderReleased()</code>
void	<code>valueChanged(int value)</code>

```
class Window : public QWidget {
private slots:
    void onSliderChanged(int value) {
        ...
    }

public:
    Window() {
        auto slider = new QSlider();
        connect(slider, &QSlider::valueChanged, this, &Window::onSliderChanged;
        ...
    }
};
```