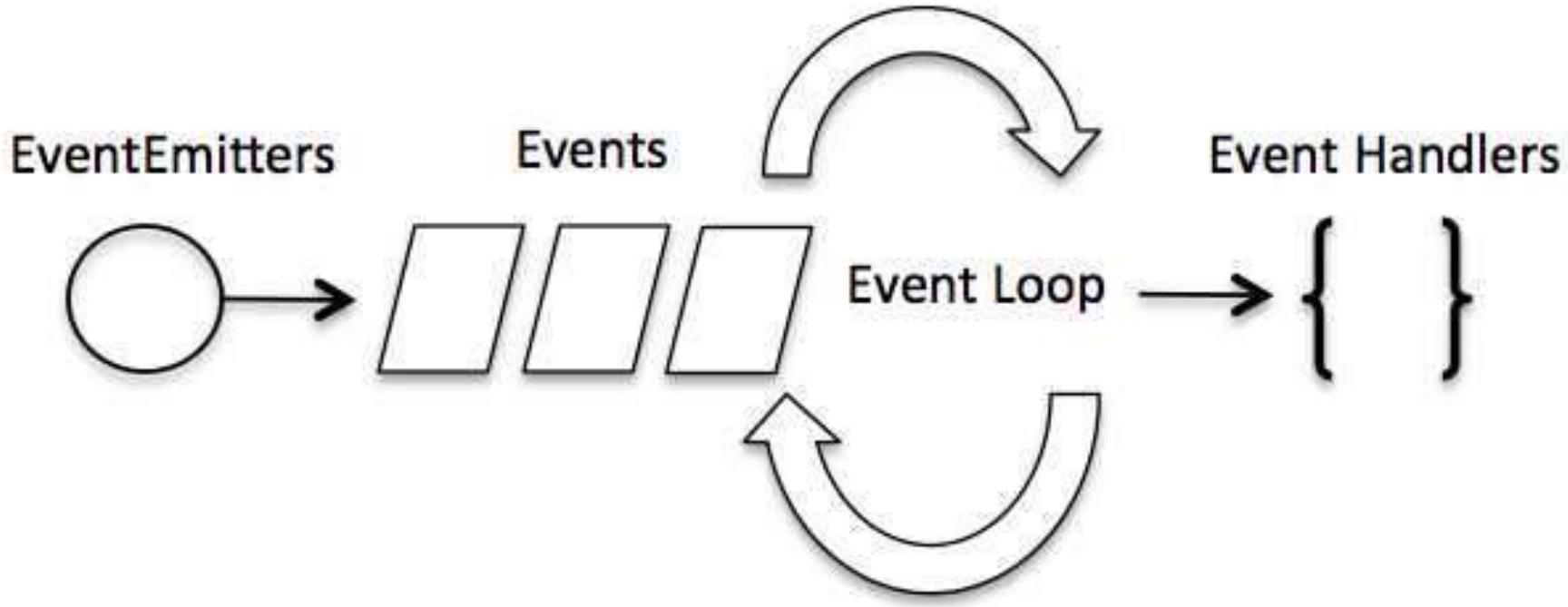
Tampereen yliopisto
Tampere University

Event-driven Programming

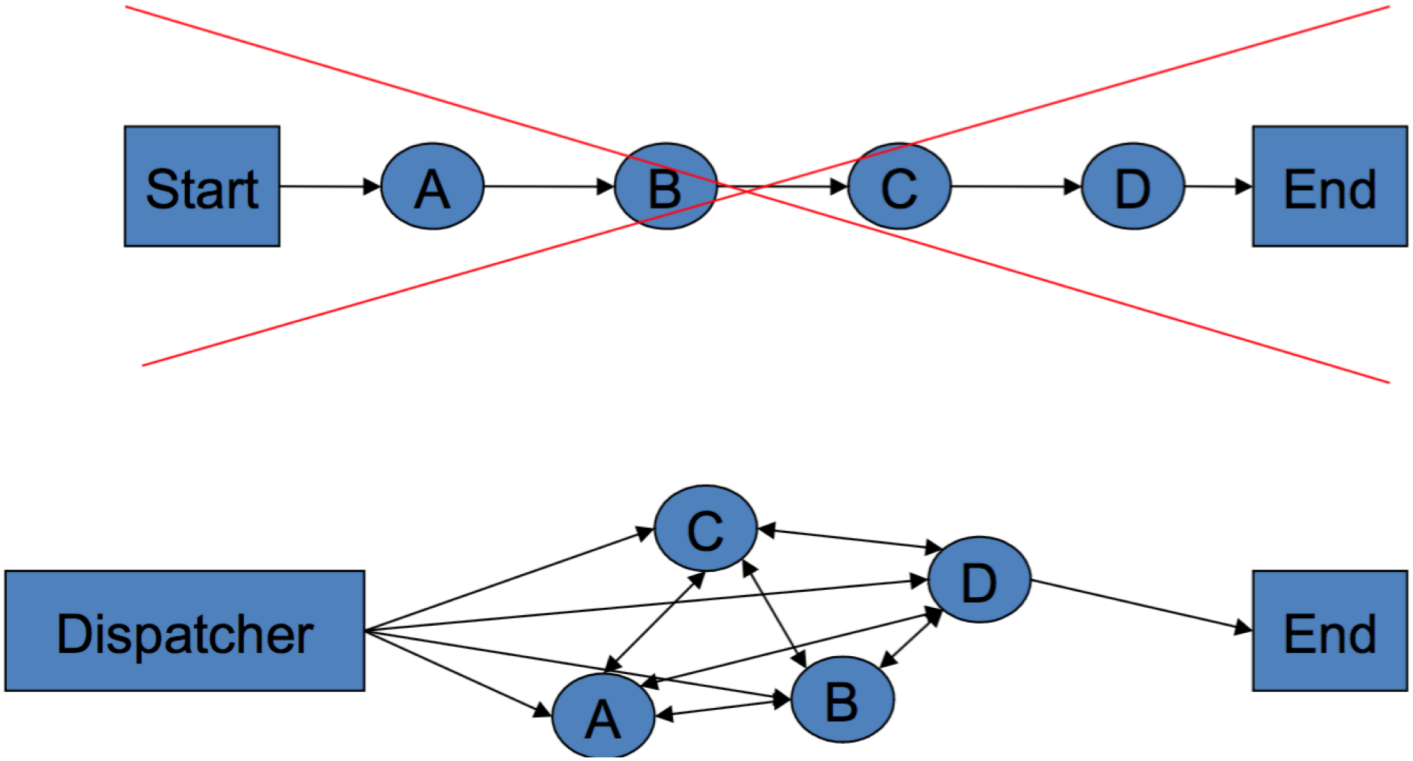
12.09.2019

Driven by events



Kuva: node.js-dokumentaatio

Traditional vs. event-driven



Events

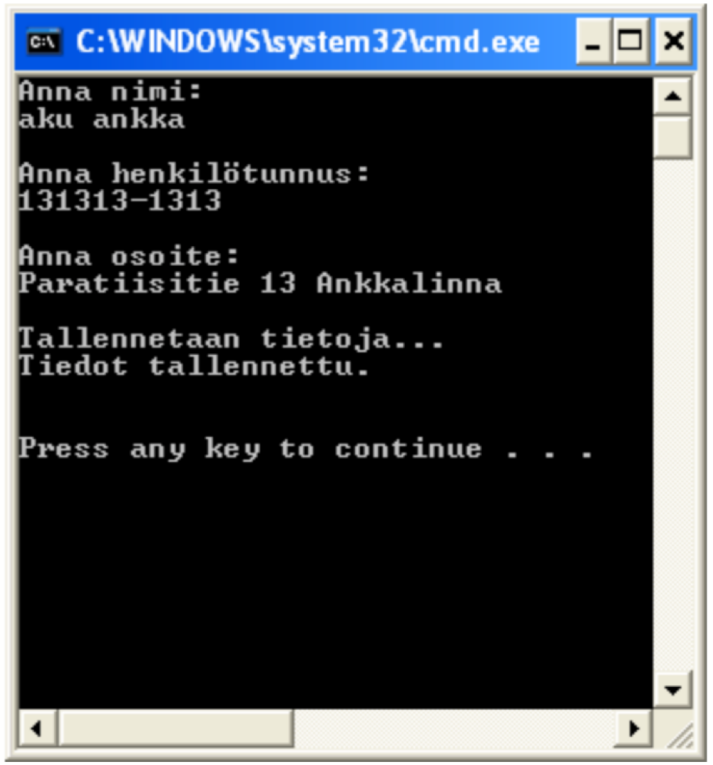
- The flow of the program is not predefined
- Events are delivered by:
  - User actions (mouse clicks, key presses)
  - Timers
  - Operating system
  - Application itself

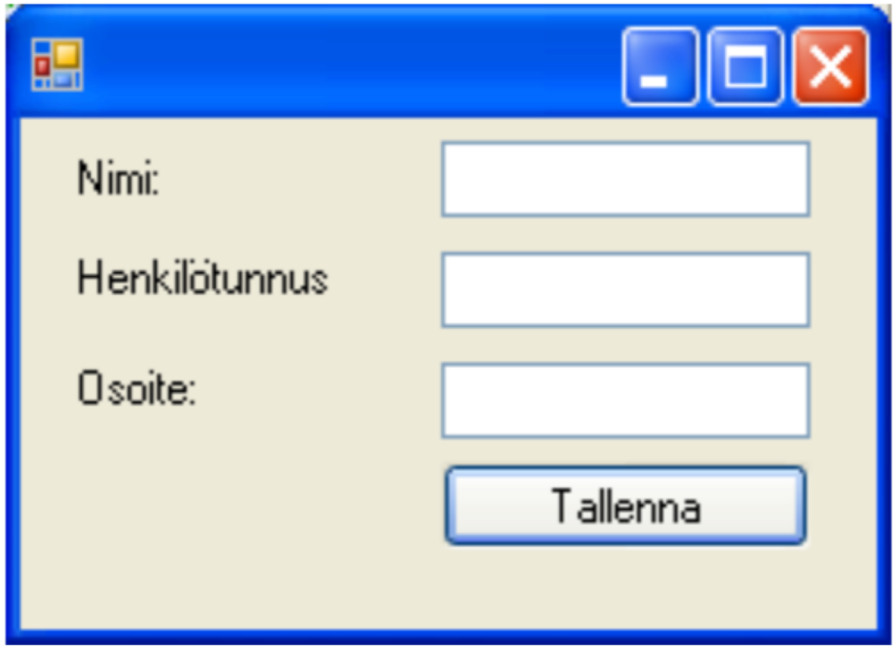
Effects to the programmer

- Program flow is not straightforward
- Program may have actions that will never be used

⇒ Testing becomes more difficult

Traditional vs. event-driven





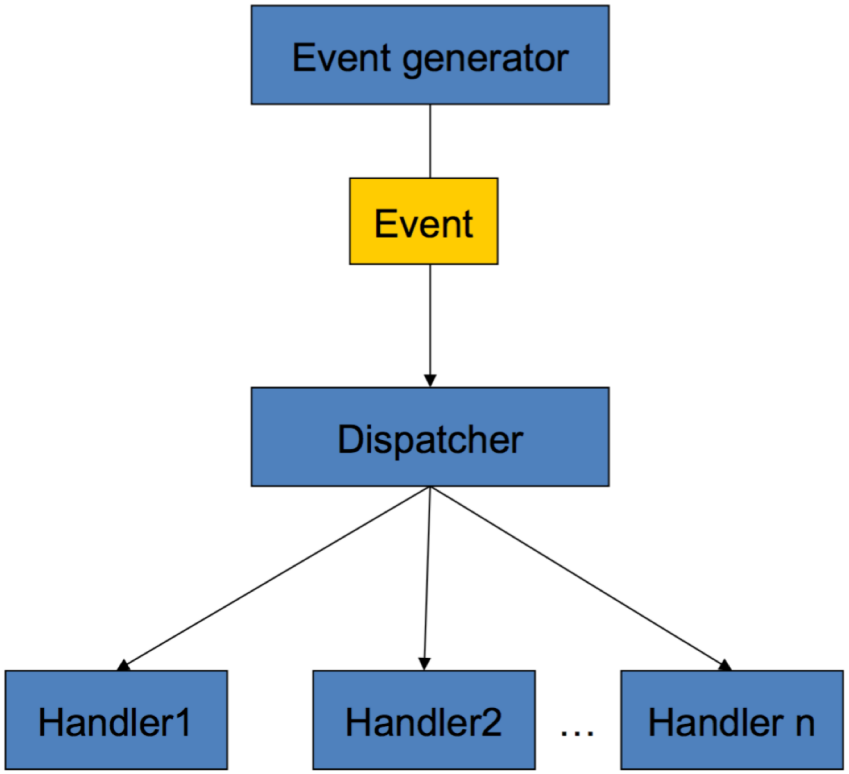
Traditional vs. event-driven

(in English)

```
void main( string[] args ) {
    string name, id, address;

    Console.WriteLine(”Give your name: ”);
    name = Console.ReadLine();

    Console.WriteLine(”Give your personal id: ”);
    id = Console.ReadLine();

    Console.WriteLine(”Give your address: ”);
    address = Console.ReadLine();
    Save(name, id, address);
}
```

```
string name, id, address;

static void main() {
    Application.Run( new Form1() );
}
private void name_handler( string text ) {
    name = text;
}
private void id_handler( string text ) {
    id = text;
}
private void address_handler( string text ) {
    address = text;
}
private void save_Click() {
    Save(name, id, address);
    Close();
}
```

Event flow

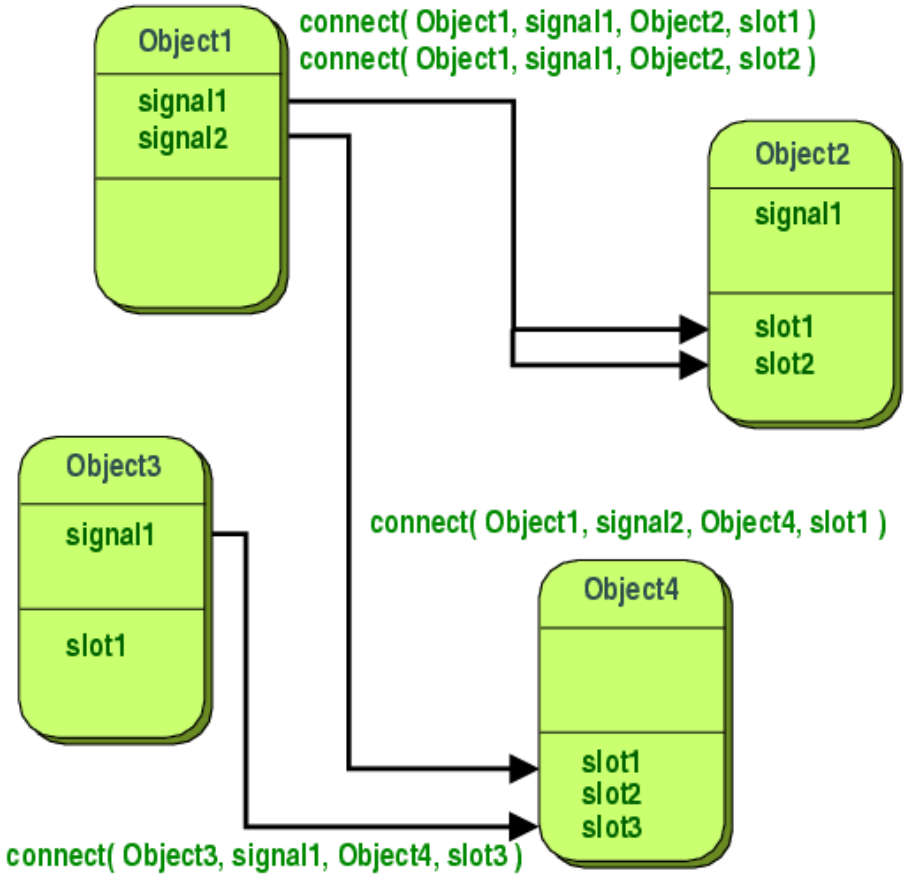

Event generator
Event
Dispatcher
Handler1
Handler2
…
Handler n

Event

- Event, signal
- Usually has no return value
- Observer (listener) is a function that is registered to listen to an event

```
connect(btnOK,SIGNAL(clicked()),
        this,SLOT(OnClicked()));
```

Qt: signals and slots



Qt: signals and slots

- Event listener is created by connecting the signal emitted by an object to the slot function
- Objects do not necessarily know each other
- Signal is emitted, when event happens ⇒ the slot function connected to the signal is called

Qt: signals and slots

- Requirements
  - A class must be a subclass of QObject (or its subclass e.g. QWidget)
  - A class definition must include Q_OBJECT macro

Example: C++ class

```
class Counter
{
public:
    Counter() { m_value = 0; }
    int value() const { return m_value; }
    void setValue(int value);
private:
    int m_value;
};
```

QObject-based class

```
class Counter : public QObject
{
    Q_OBJECT
  public:
    Counter() { m_value = 0; }
    int value() const { return m_value; }
  public slots:
    void setValue(int value);
  signals:
    void valueChanged(int newValue);
  private:
    int m_value;
};
```

Slot implementation

```
void Counter::setValue(int value)
{
    if(value != m_value) {
        m_value = value;
        emit valueChanged(value);
    }
}
```

Connecting signals and slots

```
Counter a, b;
QObject::connect(&a, &Counter::valueChanged,
                 &b, &Counter::setValue);

a.setValue(12); // a.value() == 12, b.value() == 12
b.setValue(48); // a.value() == 12, b.value() == 48
```

Qt: signals and slots

- Old syntax:

```
connect(sender,SIGNAL(valueChanged(QString,QString),
        receiver, SLOT(updateValue(QString)) );
```

- New syntax:

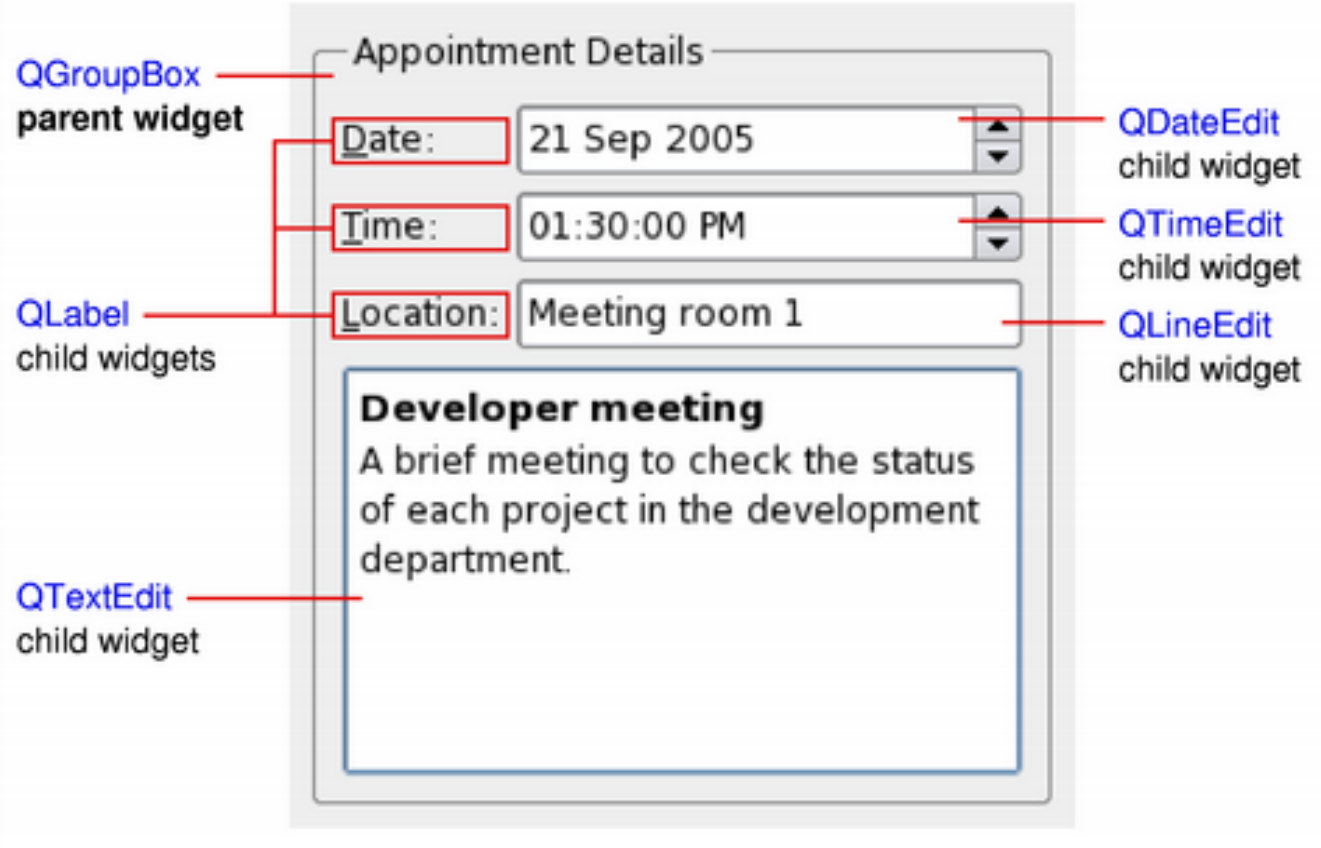
```
connect(sender, &Sender::valueChanged,
        receiver, &Receiver::updateValue );
```

Tampereen yliopisto
Tampere University

User interface

Qt Creator

UI and component layout



Component layout

- Three choices:
  - Absolute coordinates
  - Manual layout
  - Layout managers
- In Qt: also QML

Absolute coordinates

- Programmer sets the coordinates and size of the components
- User cannot resize the window
- Changing fonts may truncate text
- The components might have inappropriate sizes for some styles
- Program maintenance becomes painful

Manual layout

- Programmer still sets the positions of the components, but their sizes are calculated based on the window size
- Requires overriding resizeEvent() function of the dialog
- Result is typically better than with absolute coordinates, but implementation is still hard

Layout managers in Qt

A simple way to organize UI components automatically

- Describe the layout of the components in UI
- Automatically adjust the layout based in response to font changes, content changes, window resizing
- Ensure a regular order of the components, and keep the UI usable

Layout managers in Qt

- Rough sizes:
  - Minimum size
  - Maximum size
  - Preferred size
- Widget can be scalable, or it can have a preferred size
- Layout manager calculates the size and position of the widget whenever the layout or the size of the window changes – Resize event

Layout managers in Qt

- The easiest way to pretty layout is provided by:
  - `QHBoxLayout`
  - `QVBoxLayout`
  - `QGridLayout`, ja
  - `QFormLayout`

  which are possible to compose inside of each other

QHBoxLayout

For horizontal layout

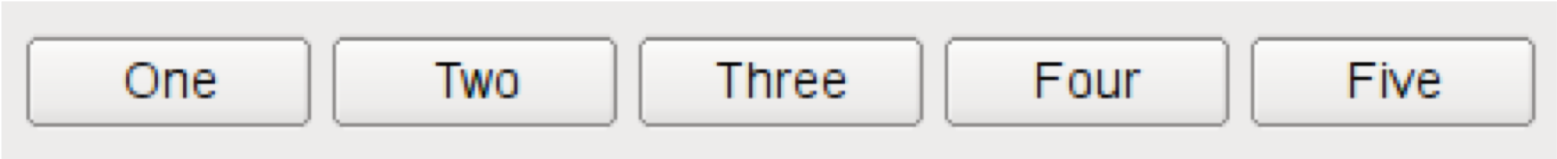


Fig: Qt documentation

QVBoxLayout

For vertical layout

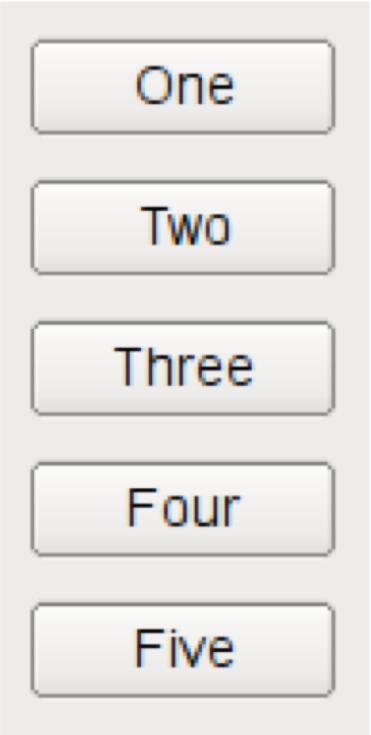


Fig: Qt documentation

QGridLayout

For grid layout

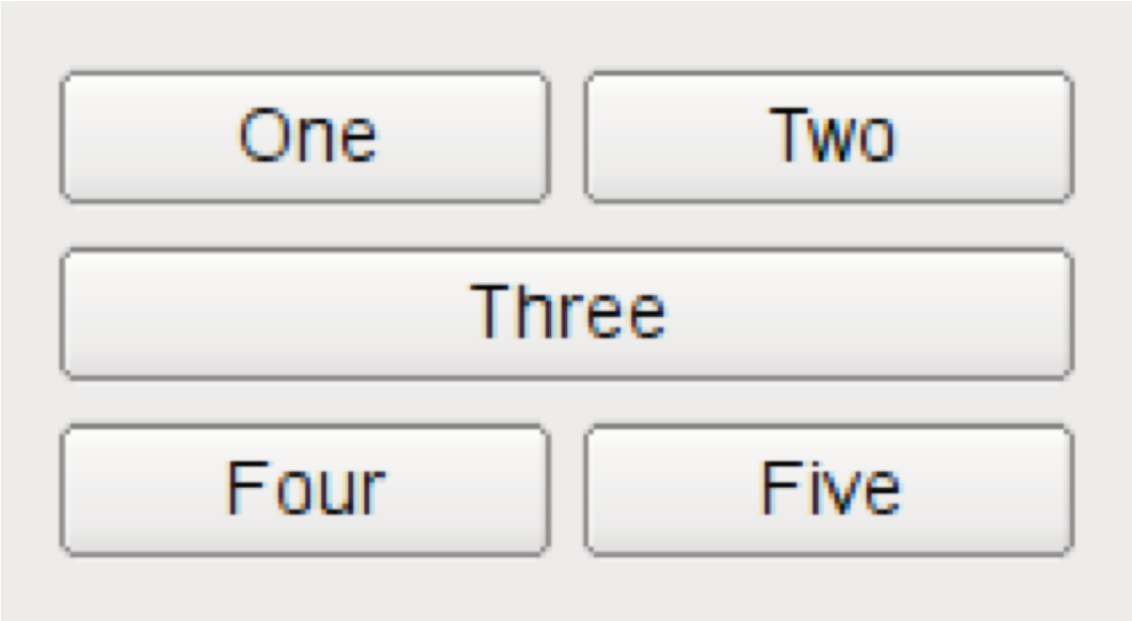


Fig: Qt documentation

QFormLayout

For two columns

Fig: Qt documentation

Nested layout

A dialog window typically consists of several nested layouts

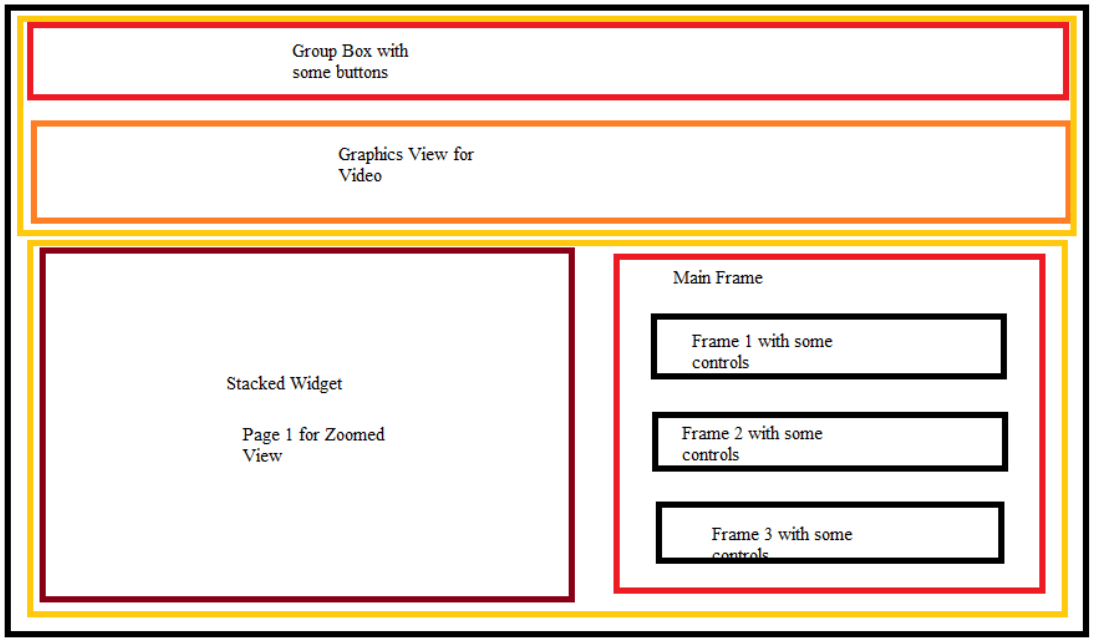


Fig: Stackoverflow

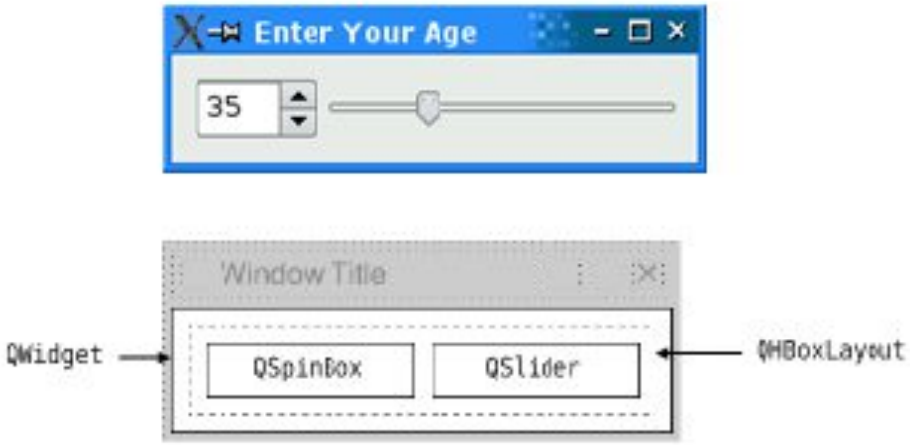
Parent

- Layout sets a parent widget (reparent) for a widget automatically
- Parent widget is not itself a layout but a widget, which has a layout set in (setLayout)

Example

- Three widgets:
  - parent: QWidget (main window)
  - children: QSpinBox and QSlider
- Signals and slots synchronize widgets

```
#include <QApplication>
#include <QHBoxLayout>
#include <QSlider>
#include <QSpinBox>

int main(int argc, char *arch[]) {
    QApplication app(archc, argv);
    QWidget *window = new QWidget;
    window->setWindowTitle("Enter your age");
    QSpinBox *spinBox = new QSpinBox;
    QSlider *slider = new QSlider(QT::Horizontal);
    spinBox->setRange(0, 130);
    slider->setRange(0, 130);              // to be
continued
```



```
    QObject::connect(spinBox,
SIGNAL(valueChanged(int)),
                        slider, SLOT(setValue(int)));
    QObject::connect(slider,
SIGNAL(valueChanged(int)),
                        spinBox, SLOT(setValue(int)));
    spinBox->setValue(35);
    QHBoxLayout *layout = new QHBoxLayout;
    layout->addWidget(spinBox);
    layout->addWidget(slider);
    window->setLayout(layout);
    window->show();
    return app.exec();
}
```

Using several windows

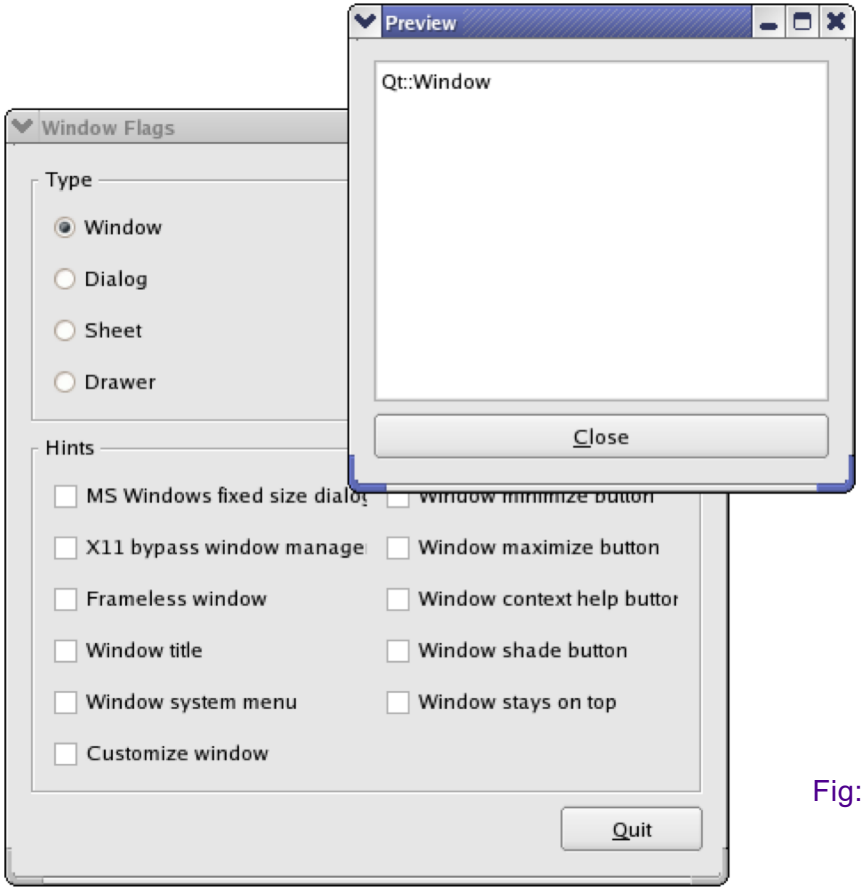


Fig: Qt documentation