Lecture Qt004
Signal-Slot Concept

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

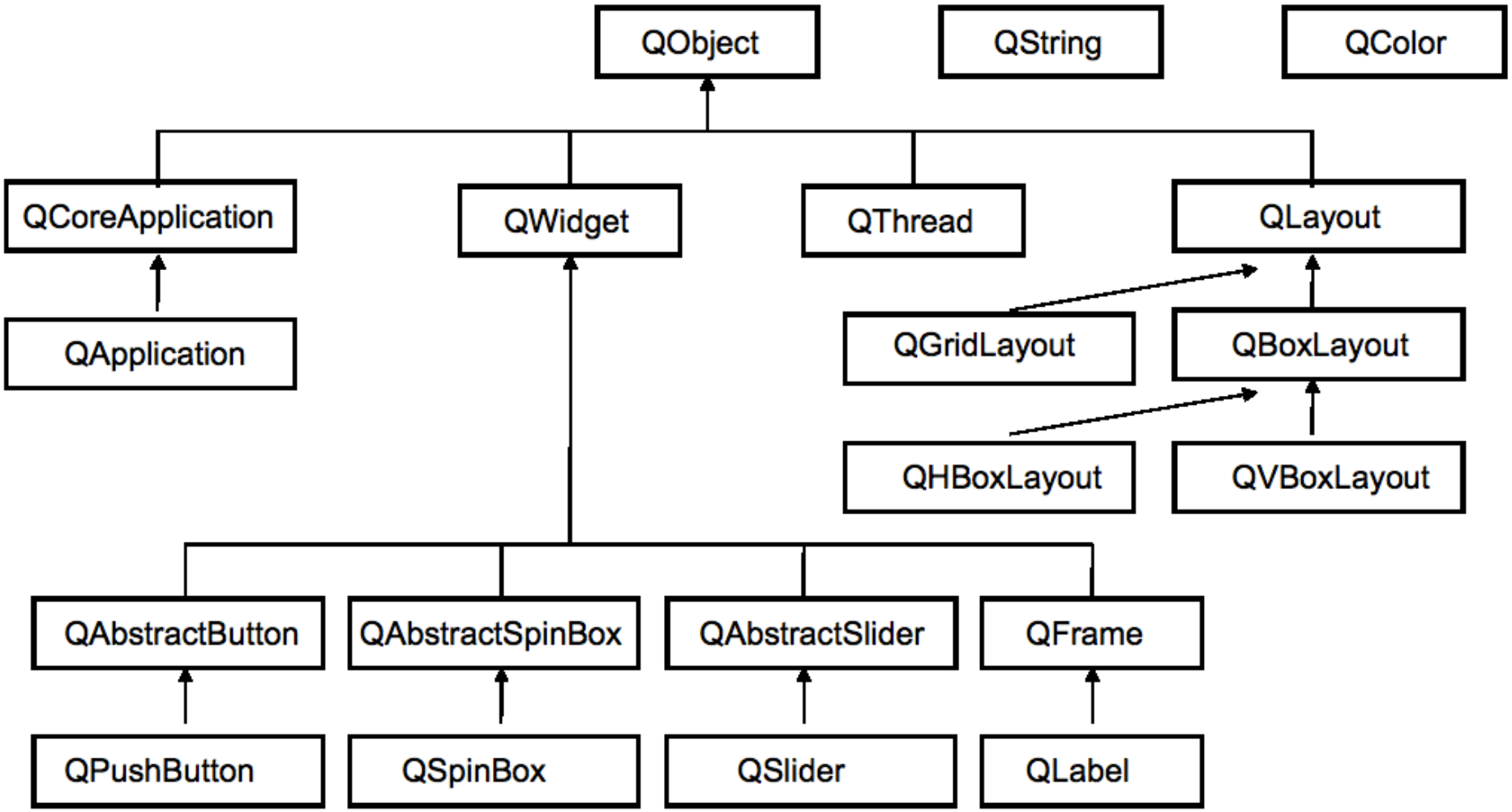
Department of Electrical and Computer Engineering

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Outline

- Qt Inheritance
- Signals, Slots, and Events
  - Qt4-style connect
  - Qt5-style connect using functions pointers
  - Qt5-style connect to non-member function
- Overloading Signals/Slots
- ***disconnect*** Method
- Custom Signals/Slots
- Key Points

Qt Inheritance



(above derived from Molkentin Figure 1.9 and Blanchette/Summerfield Figure 1.8)

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Qt Inheritance

- Important Observations
  - All widgets (visual objects) inherit from **QWidget**
  - Non-visual objects do not inherit from **QWidget**
  - Value-based object management is used for non-visual objects

```
QString  str1 = "Hello, world!";
QString  str2 = str1;
str2.replace("world", "Qt");
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Signals, Slots, and Events - 1

- Qt applications must respond to a variety of ***events***
  - User interaction with an on-screen widget is an example of an event
    - Examples: left-click of mouse, key press, drag, etc.
  - Internal events such as expiration of a timer
- Event handlers are functions that respond to events
  - Generic event handler ***event(…)*** may respond to any type of event
  - Specific event handler responds to one type of event
    - Examples: ***mousePressEvent(…)*** or ***timerEvent(…)***

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

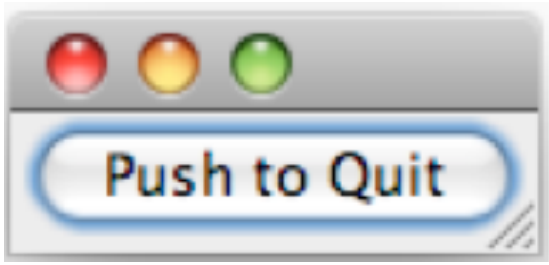
Signals, Slots, and Events - 2

- Upon the occurrence of the event, an event handler will execute and ***emit*** a signal, which will trigger execution of any connected slot functions in some order
- A ***signal*** emitted by a widget (object) indicates a user action (event) or state change
  - One signal can be connected to multiple slots within an object or multiple slots across several different objects
  - A signal can also be connected to another signal
- A ***slot*** is a **function** that automatically executes in response to connected signal that was emitted
  - All slots connected to a signal are called when that signal is emitted

Qt4-Style Signals and Slots Example 1

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

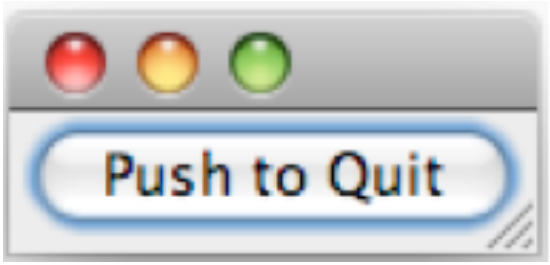
```
// pushtoquit.cpp -- Pushbutton example
#include <QApplication>
#include <QPushButton>
int main(int argc, char* argv[])
{
    QApplication    myApp(argc, argv);

    // Create push to quit button
    QPushButton pushbutton("Push to Quit");

    // Make Qt4-style signal-slot connection
    QObject::connect(&pushbutton,           // Source address
                     SIGNAL(clicked()),
                     &myApp,                // Destination address
                     SLOT(quit()));

    pushbutton.show();      // Make widget visible

    return  myApp.exec();   // Start event loop
} // End main()
```

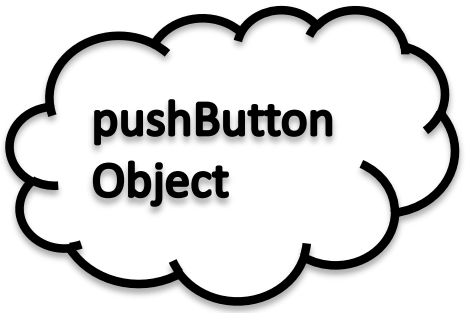


Notes:

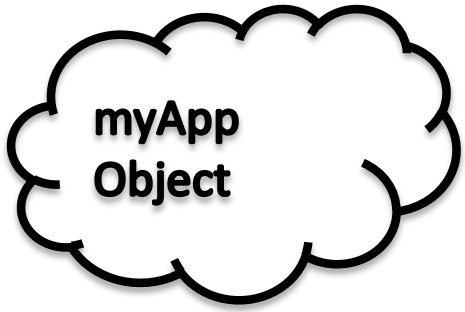
- **QtAssistant** provides a list of available signals and slots for standard Qt data types
- **SIGNAL** and **SLOT** are macros that must be used with **QObject::connect()** - more on this later
- The ***connect()*** function is a member of the ***QObject*** data type
- **::** is the scope resolution operator

Qt4-Style Signals and Slots Example 1






```
connect(  &pushButton, SIGNAL(clicked()), &myapp, SLOT(quit()) );
```




```
// Specialized event handler

void QPushButton::mousePressEvent(QMouseEvent * event)
{
   …
   emit  clicked();
   …
}
```

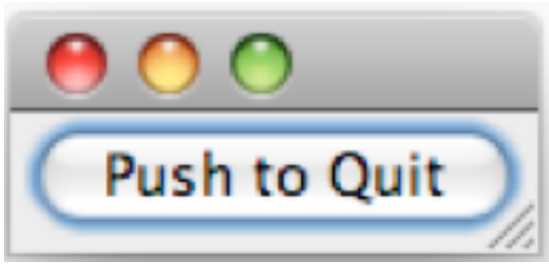
```
// Slot function

void QApplication::quit()
{
   …
}
```

We will discuss events, event handlers, and event filters is more detail later…

Qt5-Style Signals and Slots Example 2

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// pushtoquit.cpp -- Pushbutton example
#include <QApplication>
#include <QPushButton>
int main(int argc, char* argv[])
{
    QApplication    myApp(argc, argv);

    // Create push to quit button
    QPushButton pushbutton("Push to Quit");

    // Make Qt5-style signal-slot connection
    QObject::connect(&pushbutton,          // Source address
                     &QPushButton::clicked,
                     &myApp,               // Destination address
                     &QApplication::quit);

    pushbutton.show();       // Make widget visible

    return  myApp.exec();    // Start event loop
} // End main()
```



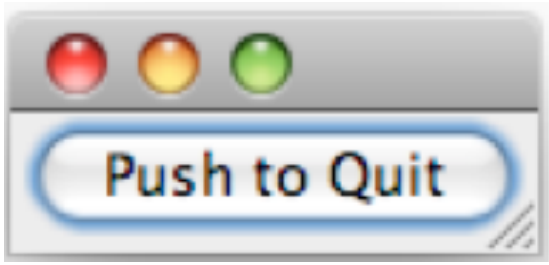
Notes:

- **Qt5-style connect** statement utilizes ***function pointers***

Qt5-Style Signals and Slots Example 3

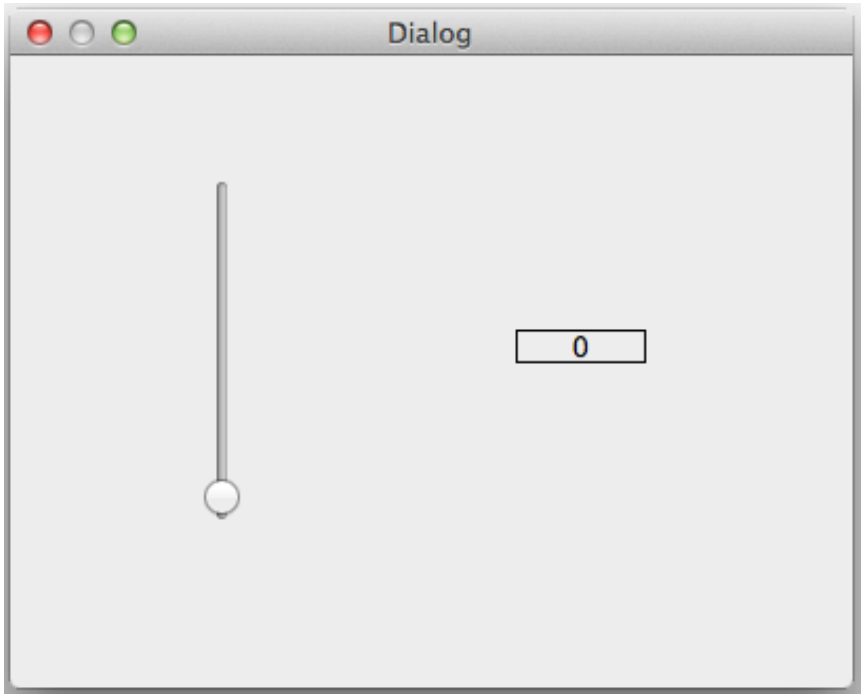
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// pushtoquit.cpp -- Pushbutton example
#include <QtDebug>
#include <QApplication>
#include <QPushButton>
static void PrintStuff()
{
  qDebug() << "HelloWorld";  // outputs to console
}

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

    // Create push to quit button
    QPushButton pushbutton("Push to Quit");

    // Make Qt5-style signal-slot connection
    QObject::connect(&pushbutton, &QPushButton::clicked,
                     &myApp, &QApplication::quit);

    // Make Qt5-style signal-slot connection to a non-member function
    QObject::connect(&pushbutton, &QPushButton::clicked,
                     &PrintStuff);

    pushbutton.show();       // Make widget visible

    return  myApp.exec();    // Start event loop
} // End main()
```



Overloading Signals/Slots - 1

- Suppose that we want user changes in the position of a slider to be reflected in text displayed by a label
- Slider values range from 0-10



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Overloading Signals/Slots - 2

- We need to identify the signal emitted by the **QSlider** object to indicate a change in the slider's current setting.
- Looking at the help page in **Qt Assistant** for **QSlider** we find that the **valueChanged** signal is emitted upon each change in current value

```
void valueChanged(int value)
```

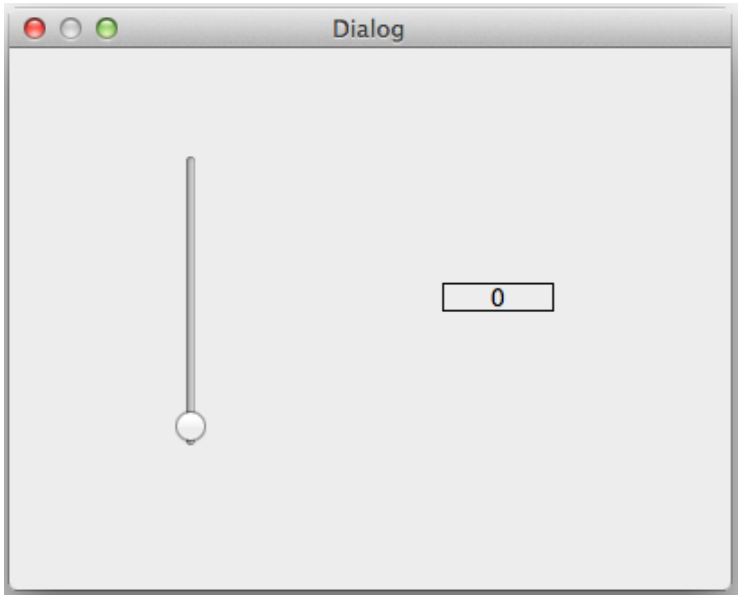
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Overloading Signals/Slots - 3

- We need to identify the slot function that we must use to modify the current contents of the **QLabel** object.
- Looking at the help page in **Qt Assistant** for **QLabel** we find that the **setText** and **setNum** slots may change the current value displayed in a **QLabel** object

```
void setText(const QString& s)
void setNum(int num)
void setNum(double num)
```

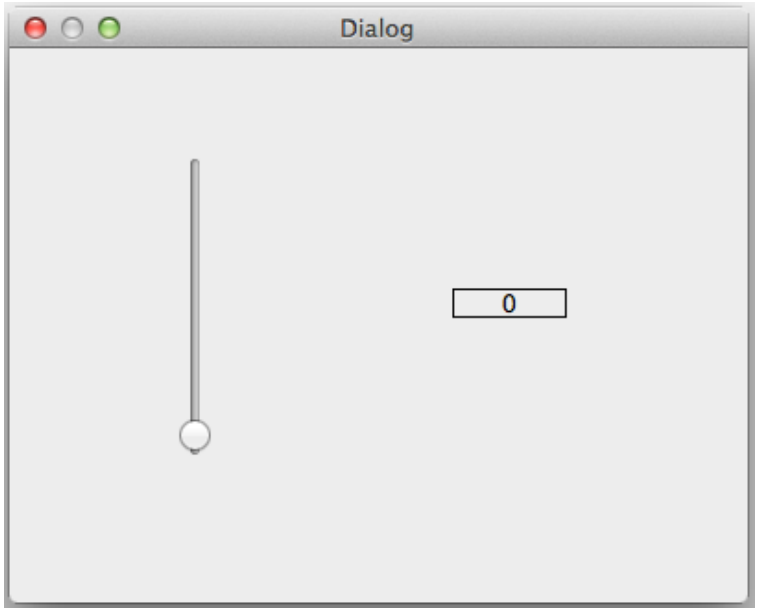
Overloading Signals/Slots - 4 connect Method – Qt4 style



Assuming that **slider** and **label** are pointers to the **QSlider** and **QLabel** objects, respectively:

```
connect( ui->slider, SIGNAL(valueChanged(int)),
         ui->label, SLOT(setNum(int)) );
```

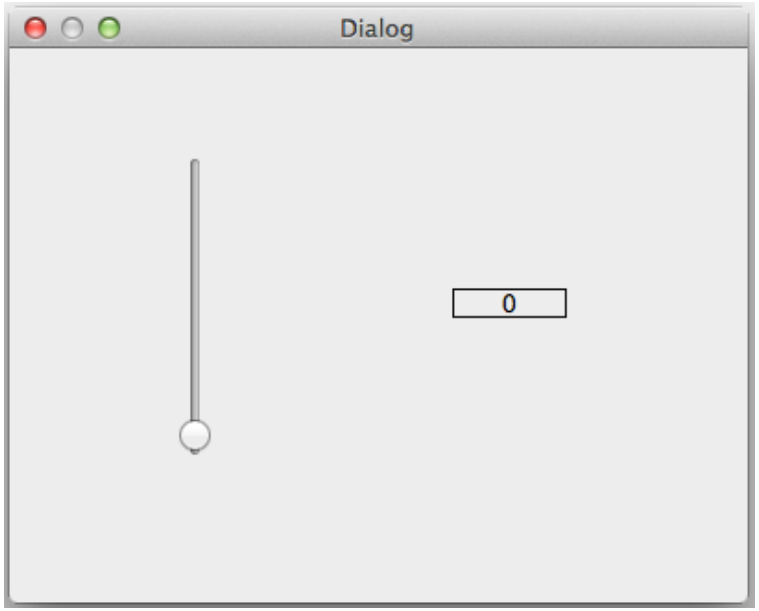
Overloading Signals/Slots - 5 connect Method – Qt5 style



Assuming that **slider** and **label** are pointers to the **QSlider** and **QLabel** objects, respectively:

```
connect( ui->slider, &QSlider::valueChanged,
         ui->label, &QLabel::setNum );
// Compile error - overloaded slot function (see QLabel in QtAssistant)
// void  setNum(int num)
// void  setNum(double num)
```

Overloading Signals/Slots - 6 connect Method – Qt5 style



Assuming that **slider** and **label** are pointers to the **QSlider** and **QLabel** objects, respectively:

```
connect( ui->slider, &QSlider::valueChanged, ui->label,
    static_cast<void (QLabel::*)(int)>(&QLabel::setNum)  );

// Type casting may be used for overloaded signals or slots
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

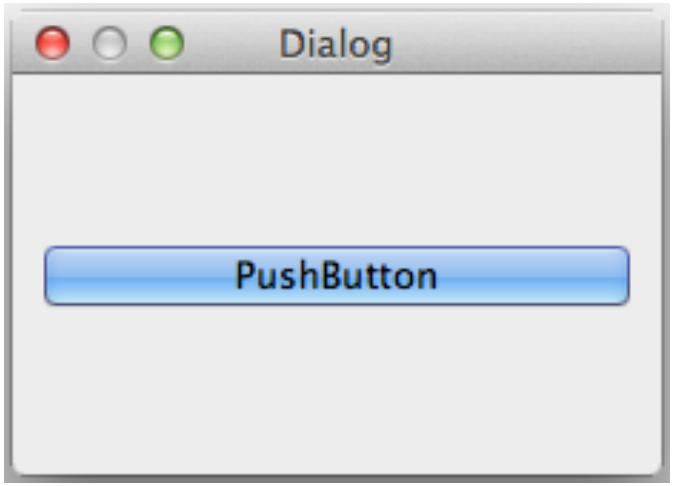
disconnect Method

- To break an existing signal-slot connection, use the reserved word **disconnect** in place of **connect**
  - Be consistent with the approach used to establish the signal-slot connection:

    Qt4-style vs Qt5-style

Custom Signals/Slots - 1

- Until now we have seen how to connect signals and slots defined with Qt data types
- We will now define a custom signal and a custom slot
- Pushing the button will trigger the custom slot **bar()**
- The slot **bar()** will emit the custom signal **foo()**



THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Custom Signals/Slots - 2

```
// main.cpp
#include "dialog.h"
#include <QApplication>
#include <QtDebug>
static void PrintStuff()
{
    qDebug() << "*** foo() ***";
}
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Dialog w;
    QObject::connect(&w, &Dialog::foo, &PrintStuff);
    w.show();
    return a.exec();
}
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Custom Signals/Slots - 3

```
// dialog.h
#ifndef DIALOG_H
#define DIALOG_H
#include <QDialog>
namespace Ui {class Dialog;}
class Dialog : public Qdialog
{
    Q_OBJECT
public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();
private:
    Ui::Dialog *ui;
signals:
    void foo();
public slots:
    void bar();
};
#endif // DIALOG_H
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Custom Signals/Slots - 4

```
// dialog.cpp
#include <QtDebug>
#include "dialog.h"
#include "ui_dialog.h"
Dialog::Dialog(QWidget *parent) : QDialog(parent), ui(new Ui::Dialog)
{
   ui->setupUi(this);
   connect(ui->pushButton, &QPushButton::clicked, this, &Dialog::bar);
}
Dialog::~Dialog()
{
    delete ui;
}
void Dialog::bar()
{
    qDebug() << "*** bar() ***";
    emit foo();
}
```

Key Points

- Qt's signal-slot mechanism provides a convenient means of linking events such as user-interactions with graphical objects to code that must respond
- Qt4-style connections via SIGNAL/SLOT macros and Qt5-style connections via function pointers each have their advantages and disadvantages