# Можно задать любую степень прозрачности окна посредством:

void setWindowOpacity(qreal level);

# Курсор

Изменить вид курсора:

void QWidget::**setCursor** ( const QCursor& );

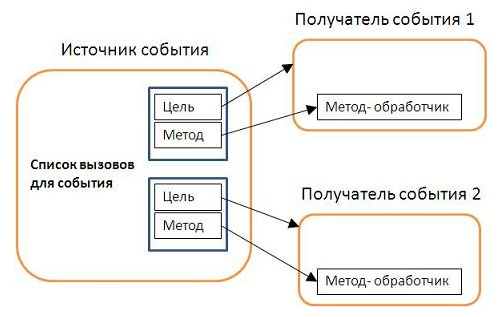
Восстановить умолчание – посредством:  
void QWidget::**unsetCursor** ();

## Свой курсор

QPixmap pxm("Images/spider1.bmp");

pxm.setMask(QBitmap("spider1.bmp"));   
QCursor curs(pxm);   
setCursor(curs);

# События (events) в Qt



## События клавиатуры

virtual void *keyPressEvent* (QKeyEvent\* event) - вызывается при нажатии одной из клавиш на клавиатуре

virtual void *keyReleaseEvent* (QKeyEvent\* event) - вызывается при отпускании клавиши на клавиатуре

int QKeyEvent::**key()** const – позволяет определить, какая клавиша инициировала событие.

void Widget::*keyPressEvent*(QKeyEvent \*event)

{

if (event->key() == Qt::Key\_Left)

{

x1-=10;

x2-=10;

}

…

repaint();

}

## События мыши

virtual void *mousePressEvent* (QMouseEvent\* event) - вызывается при нажатии одной из клавиш мыши

virtual void *mouseReleaseEvent* (QMouseEvent \* event) - вызывается при отпускании клавиши мыши

virtual void *mouseMoveEvent* (QMouseEvent \* event) - вызывается при перемещении указателя мыши, только если одна из ее кнопок нажата

Координаты курсора мыши относительно левого верхнего угла окна (клиентские координаты). Методы  
int QMouseEvent::x()const  
и int QMouseEvent::y()const  
или const QPoint& QMouseEvent::pos () const

Какая из кнопок мыши нажата. Для получения этой информации можно использовать метод  
[Qt::MouseButtons](qthelp://org.qt-project.qtgui.532/qtcore/qt.html) QMouseEvent::button() const

void MainWindow::mousePressEvent (QMouseEvent \* event)

{

if (event->button() == Qt::LeftButton)

qDebug()<< "Xpos = " << event->x();

}

## Событие таймера

virtual void *timerEvent* (QTimerEvent\* event) ;- обработчик события

int QObject::startTimer(int*interval*, [Qt::TimerType](qthelp://org.qt-project.qtcore.532/qtcore/qt.html)*timerType* = Qt::CoarseTimer);

void QObject::killTimer(int*id*)

# Метод QWidget::repaint()

Вызывает немедленную перерисовку окна

# Класс QString

## Конструктор

**QString::**QString**(const char \* *str*)**

## Получение указателя const char\* из QString

QString qstr;

//сформировали строку

**std::string str = qstr.**toStdString**();**

const char\* p = str.c\_str();

## Преобразование int в QString

QString & QString::setNum(int *n*, int *base* = 10)

**QString QString::**number**(int *n*, int *base* = 10) [static]**

## Преобразование из QString в int:

**int QString::**toInt**(bool \* *ok* = 0, int *base* = 10) const**

## Основные действия со строками

### Сравнение

QString str = "Beatles";

bool b1 = (str == "Beatles"); // b1 = true

bool b2 = (str != "Beatles"); // b2 = false

### Длина строки

QString str = "Beatles";

int n = str.length();

### Объединение строк

QString str1 = "Be";

QString str2 = "atles";

QString str3 = str1 + str2; // str3 = "Beatles"

str1.append(str2); //str1 = "Beatles"

# QStringList – массив строк

QStringList strList;

strList << "Ringo" << "Star";

## Доступ к строкам в QStringList

for (int i = 0; i < strList.size(); ++i)

{

QString temp = str.**at**(i);

}

## Объединение строк

QString str = strList.**join**(" ");

// str == "Ringo Star";

## Разбиение на строки

QString str = "C++, Java, C#";

QStringList list;

list = str.**split**(",");

// list.at(0) = C++;

// list.at(1) = Java;

// list.at(2) = C#;