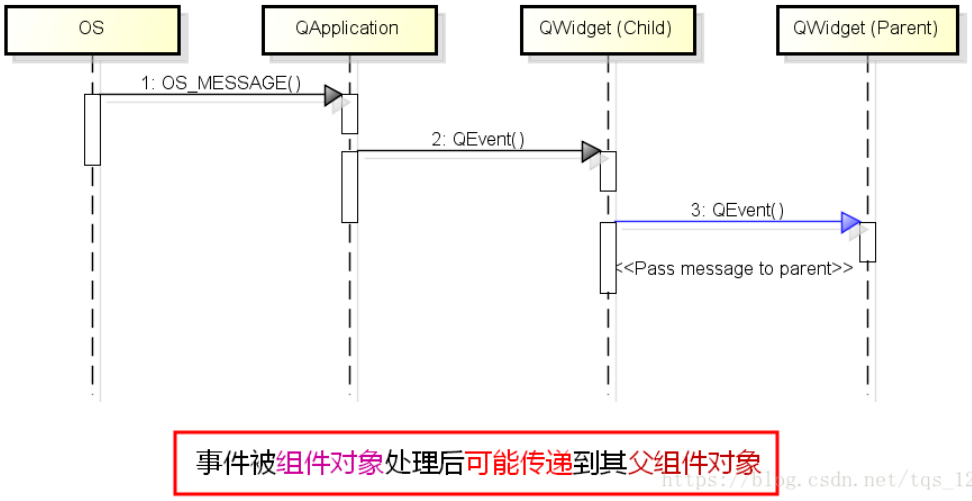
事件传递过程

- 事件的传递过程



事件被组件对象处理后可能传递到其父组件对象

在子组件中将事件进行处理后可能再将事件传递给父组件对象。

QEvent类

QEvent类是所有事件处理的父类

关键成员函数

- void ignore();接收者忽略当前事件，但事件可能传递给父组件
- void accept();接收者期望处理当前事件
- bool isAccept();判断当前事件是否被处理

示例：事件处理的顺序

先构建一个自定义的MyLineEdit类，在类里重写event()和keyPressEvent()

- bool MyLineEdit::event(QEvent* event)
- void MyLineEdit::keyPressEvent(QKeyEvent* event)

```
bool MyLineEdit::event(QEvent* event)
{
 if( event->type() == QEvent::KeyPress )
 {
 qDebug() << "MyLineEdit::event";
```

```
 }
 return QLineEdit::event(event);//调用默认事件处理函数
}
void MyLineEdit::keyPressEvent(QKeyEvent* event)//键盘按键事件
{
 qDebug() << "MyLineEdit::keyPressEvent";
 QLineEdit::keyPressEvent(event);
 e->ignore();//当前对象忽略处理此事件，所以父组件对象进行事件处理函数的调用
}
```

- Widget作为父组件，当子组件忽略事件处理时将会将事件传递到父组件进行处理。

```
bool Widget::event(QEvent* event)
{
 if(e->type() == QEvent::KeyPress)
 {
 qDebug() << "Widget::event";
 }
 return QWidget::event(event);
}
void Widget::keyPressEvent(QKeyEvent* event)
{
 qDebug() << "Widget::keyPressEvent";
 QWidget::keyPressEvent(event);
}
```

- **重写事件处理函数后，由系统发送来的系统消息就会由重写后的event函数处理，然后根据事件类型调用相应的事件处理函数。**
- **当事件类型是按下键盘时触发调用MyLineEdit::keyPressEvent函数，由于调用了ignore()函数，它就会告诉应用程序当前事件没有被处理，所以就会传递给父组件去处理。**

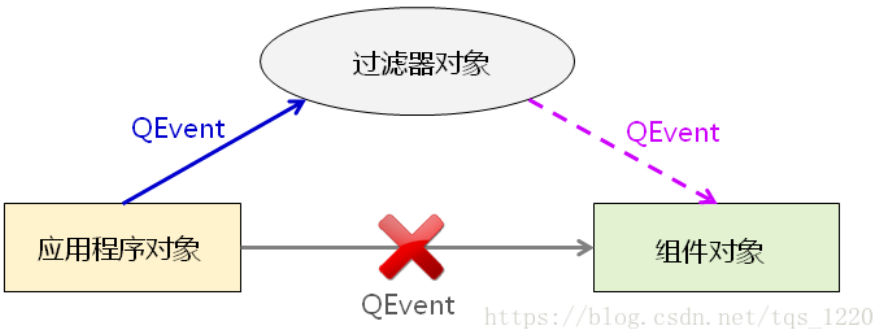
```
MyLineEdit::event
MyLineEdit::keyPressEvent
Widget::event
Widget::keyPressEvent
```

在执行了子组件（MyLineEdit对象）的事件处理函数时后又执行了父组件（Widget对象）的事件处理函数。

注意：如果我们希望父组件对象或其他来接着处理当前的事件，那么一定要调用ignore()函数。

事件过滤器

- 事件过滤器可以对其他组件接收到的事件进行监控。
- 任意的QObject对象都可以作为事件过滤器使用，QWidget对象也是一个QObject对象。
- 事件过滤器需要重写eventFilter()函数。
- 监控的意义在于可以将事件没收，用来定制一些有用的GUI效果。
- 事件过滤器特点：
  - 组件通过 installEventFilter() 函数安装事件过滤器
    - 事件过滤器在组件之前接收到事件
    - 事件过滤器能够决定是否将事件转发到组件对象



这里写图片描述

- 事件过滤器使用套路：判断某个对象，判断某种事件类型，调用默认处理函数。

- 事件过滤器的典型实现

```
// 返回 true 表示事件已经处理，无需传递给 obj，
// 返回 false 则正常传递到 obj
bool Widget::eventFilter(QObject* obj, QEvent* e)
{
    if( /* 根据 obj 判断对象 */ )
    {
        if( /* 根据 e->type() 判断事件 */ )
        {
            /* 事件处理逻辑 */
        }
    }

    /* 调用父类中的同名函数 */
    return QWidget::eventFilter(obj, e);
}
```


这里写图片描述

示例：事件过滤

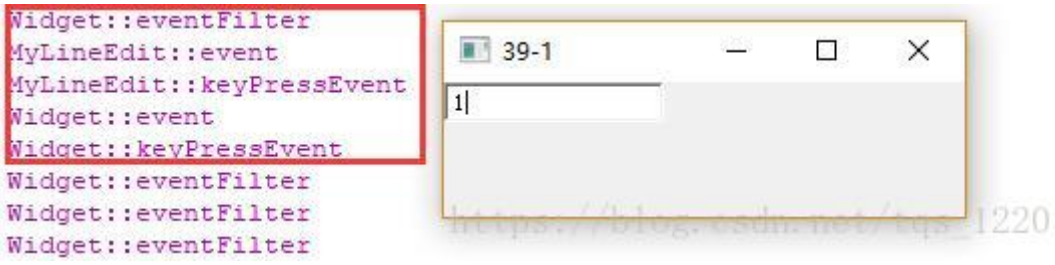
事件管理器可以对其它组件接收到的事件进行监控，事件过滤器能决定是否将事件转发到组件对象

只有按下的是数字，事件才会被传递给安装了事件过滤器的组件对象。

- 两个判断：判断满足条件的事件，判断要监控的组件对象
- 在使用事件过滤器之前一定要先安装过滤器到想要关注的组件对象，installEventFilter()安装事件过滤器
- 只有当按下的是数字才会将事件发送到组件对象（当eventFilter返回false时发送事件到组件对象）。否则会将其没收，不会进行事件处理，

```
1  Widget::Widget(QWidget *parent): QWidget(parent), LineEdit(this)
2  {
3   LineEdit.installEventFilter(this);//安装事件过滤器
4  }
5
6  bool Widget::eventFilter(QObject* obj, QEvent* e)
7  {
8   bool ret = true;//默认没收事件
9   //判断是否时要监控的对象，以及是否是要监控的事件
10   if( (obj == &LineEdit) && (e->type() == QEvent::KeyPress) )
```

```
{
qDebug() << "Widget::eventFilter";
QKeyEvent* evt = dynamic_cast<QKeyEvent*>(e);
switch(evt->key())//只有数字才会返回flase被传递到组件对象
{
case Qt::Key_0:
case Qt::Key_1:
case Qt::Key_2:
case Qt::Key_3:
case Qt::Key_4:
case Qt::Key_5:
case Qt::Key_6:
case Qt::Key_7:
case Qt::Key_8:
case Qt::Key_9:
ret = false;
break;
default:
break;
}
}
else
{
ret = QWidget::eventFilter(obj, e);
}
return ret;
}
```



这里写图片描述