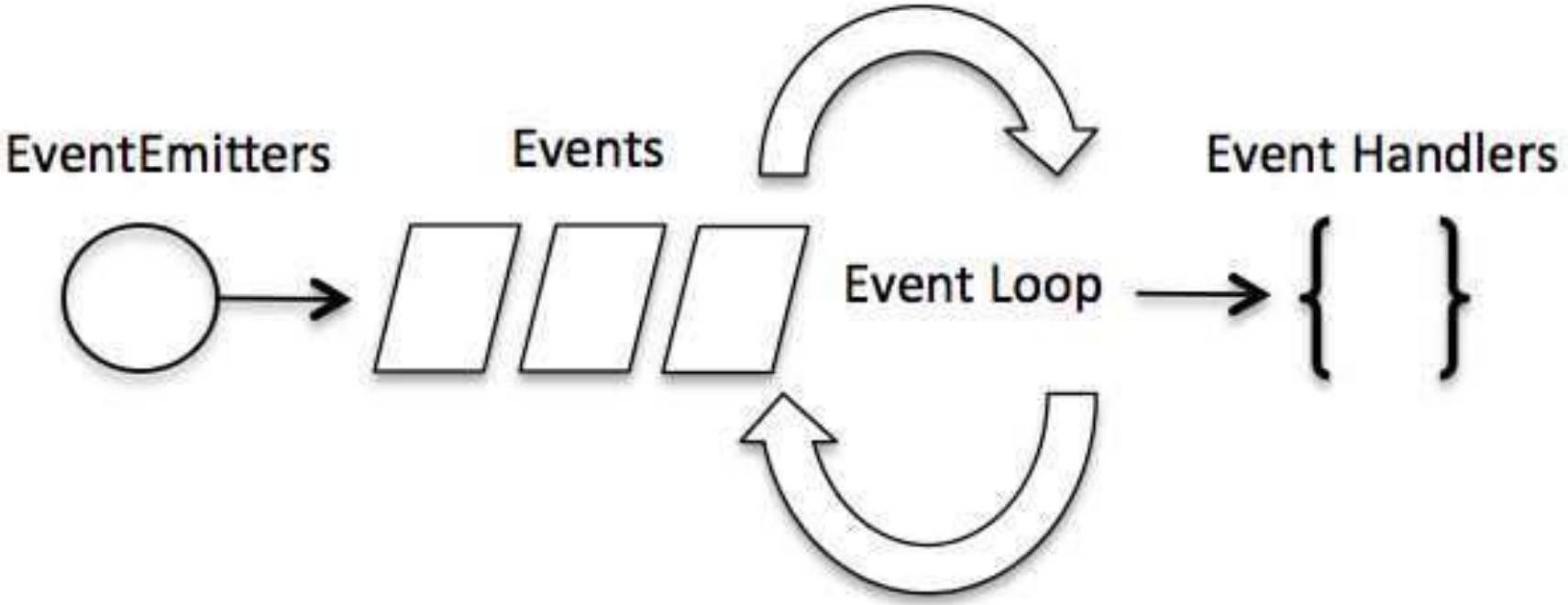
Tampereen yliopisto
Tampere University

Tapahtumapohjainen ohjelmointi

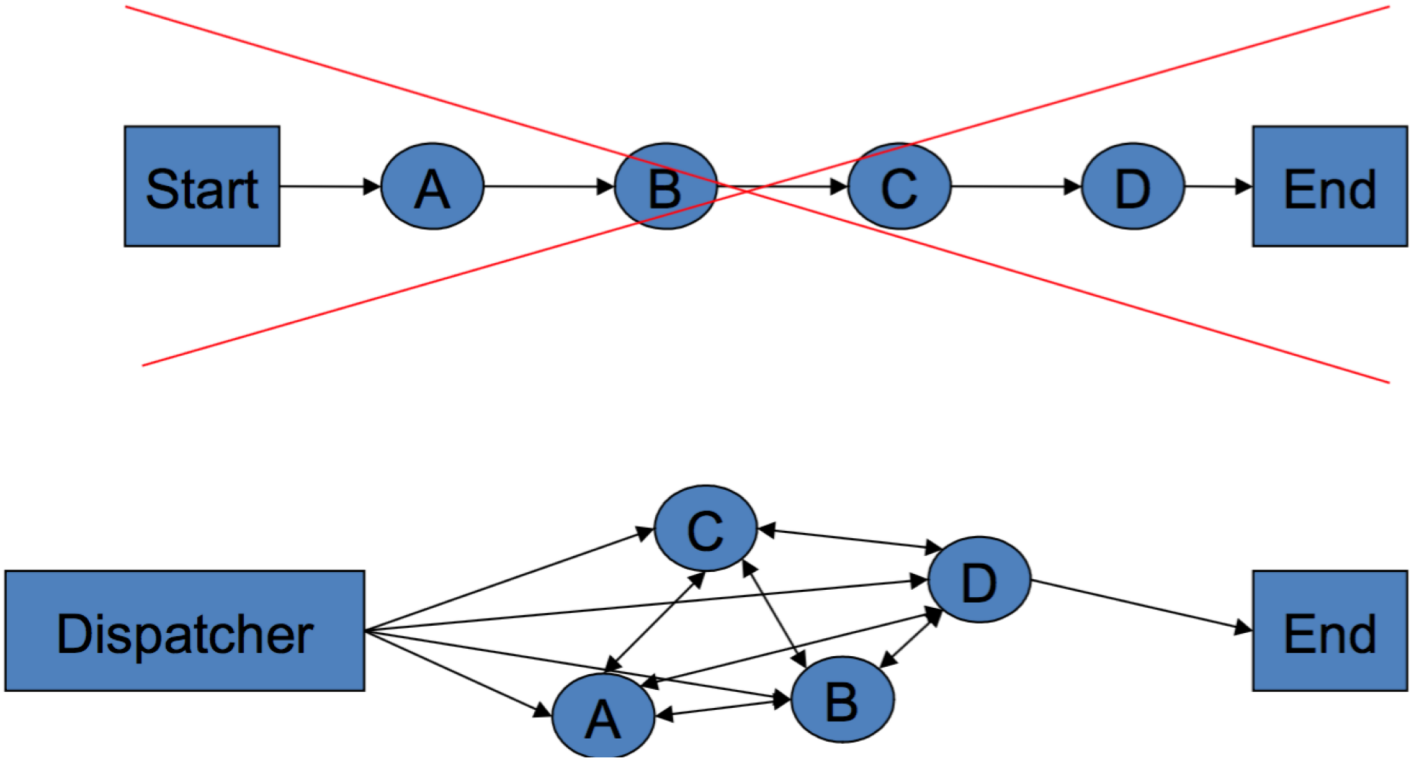
Qt Creator

Tapahtumapohjaisuus



Kuva: node.js-dokumentaatio

Perinteinen vs. tapahtumapohjainen



Tapahtumat

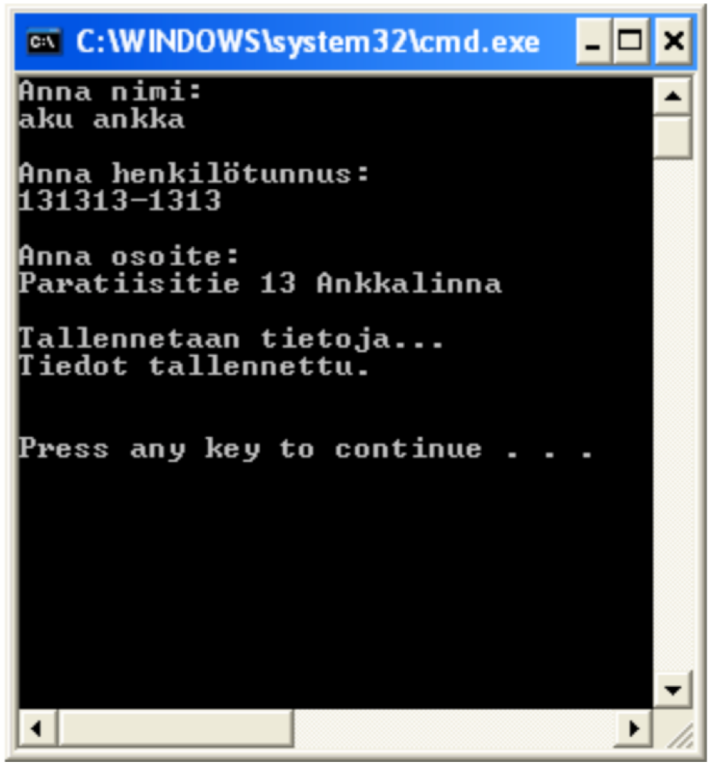
- Sovelluksen kulku ei ennalta tiedossa
- Tapahtumia synnyttävät:
  - Käyttäjän toimet (hiiren- ja näppäimistön painallukset)
  - Ajastimet
  - Käyttöjärjestelmä
  - Sovellus itse

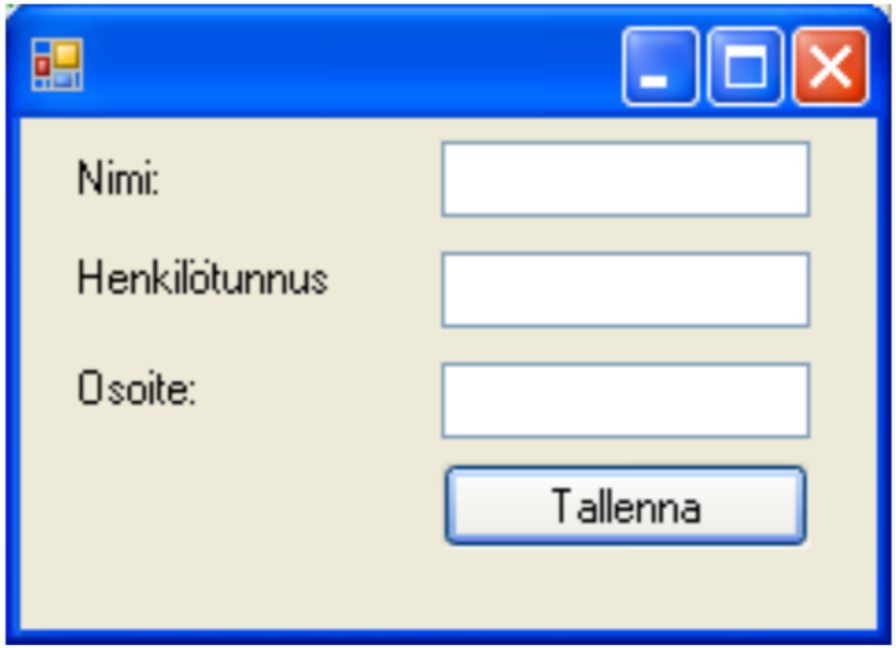
Vaikutus koodarin elämään

- Ei suoraviivaisesti etenevää suoritusta
- Toimintoja, joita ei ehkä koskaan käytetä

⇒ Testaaminen vaikeutuu

Perinteinen vs. tapahtumapohjainen





Tampereen yliopisto
Tampere University

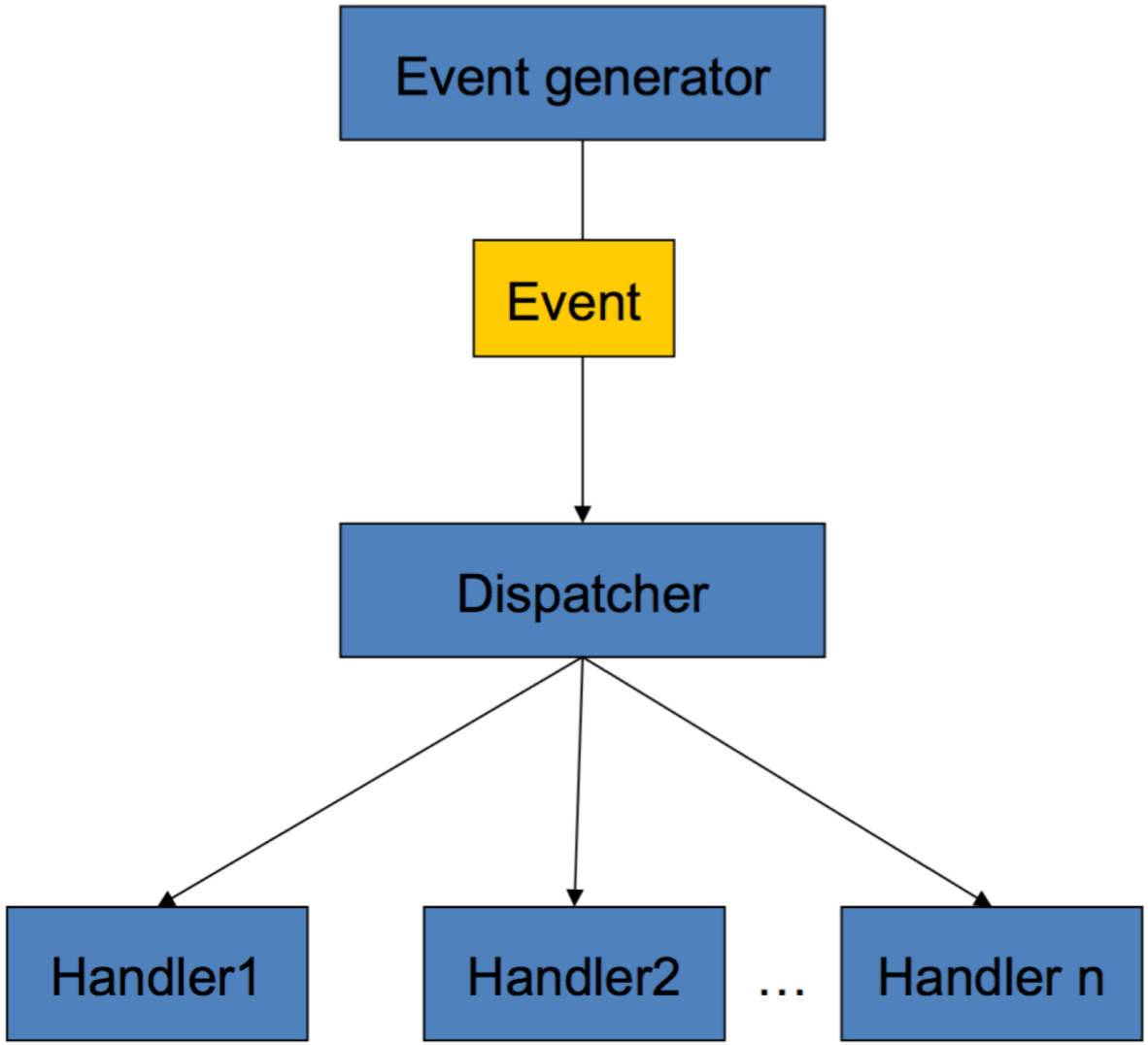
Perinteinen vs. tapahtuma-pohjainen

```
void main( string[] args ) {
    string nimi, hetu, osoite;

    Console.WriteLine(”Anna nimi: ”);
    nimi = Console.ReadLine();

    Console.WriteLine(”Anna henkilötunnus: ”);
    hetu = Console.ReadLine();

    Console.WriteLine(”Anna osoite: ”);
    osoite = Console.ReadLine();
    Save(nimi, hetu, osoite);
}
```

```
string nimi, hetu, osoite;

static void main() {
    Application.Run( new Form1() );
}
private void nimi_handler( string text ) {
    nimi = text;
}
private void hetu_handler( string text ) {
    hetu = text;
}
private void osoite_handler( string text ) {
    osoite = text;
}
private void tallenna_Click() {
    Save(nimi, hetu, osoite);
    Close();
}
```

Tampereen yliopisto
Tampere University

Tapahtuman kulku

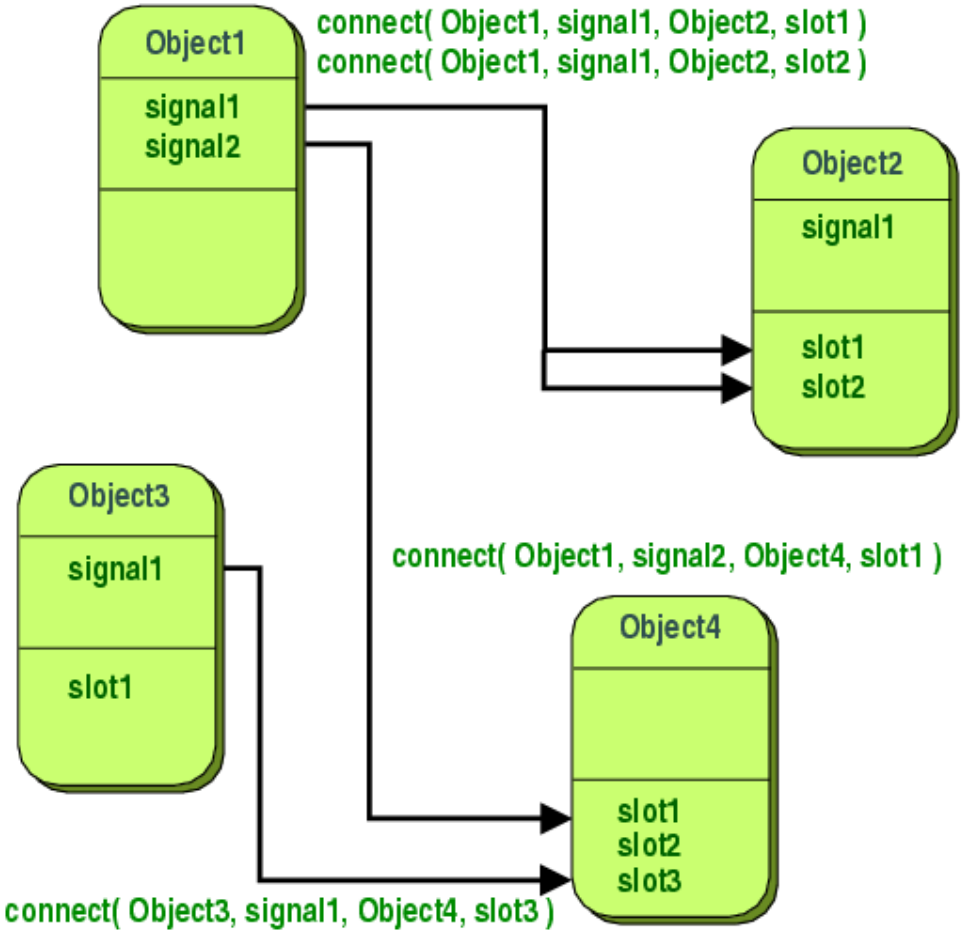


Tapahtuma

- Event, signal
- Ei yleensä paluuarvoa
- Kuuntelijat funktioita, jotka rekisteröidään kuuntelemaan tapahtumaa

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

Qt: signaalit ja slotit



Qt: signaalit ja slotit

- Tapahtumakuuntelija luodaan kytkemällä olion lähettämä signaali slot-funktioon
- Olioiden ei tarvitse tuntea toisiaan
- Signaali lähetetään, kun tapahtuma "tapahtuu" ⇒ signaaliin kytkettyä slot-funktiota kutsutaan

Qt: signaalit ja slotit

- Vaatimuksia
  - Luokkien tulee periytyä QObject:sta (tai sen aliluokasta, esim. QWidget)
  - Esittelyssä tulee olla Q_OBJECT-makro

Esimerkki: C++-luokka

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

Tampereen yliopisto
Tampere University

QObject-luokan aliluokka

```
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;
};
```

Slotin toteutus

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

Tampereen yliopisto
Tampere University

Signaalin kytkeminen slotiin

```
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: signaalit ja slotit

- Vanha syntaksi:

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

- Uusi syntaksi:

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

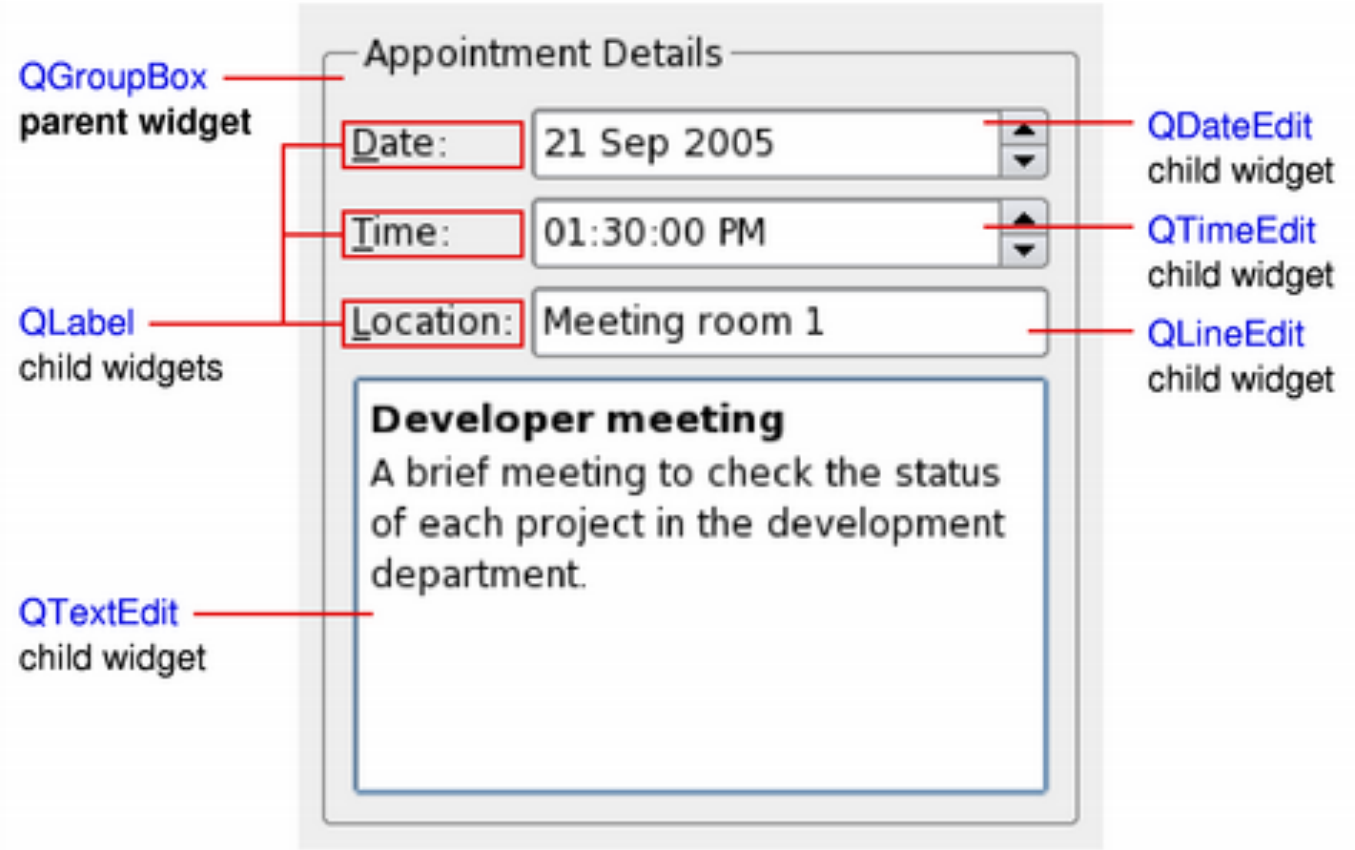
Tampereen yliopisto
Tampere University

Käyttöliittymä

11.09.2019

Tampereen yliopisto
Tampere University

Käyttöliittymä ja komponenttien asemointi



Komponenttien asemointi

- Kolme vaihtoehtoa:
  - Absoluuttiset koordinaatit
  - Manuaalinen layout
  - Layout-managerit
- Qt:ssa myös QML

Absoluuttiset koordinaatit

- Ohjelmoija määrittää jokaisen komponentin koordinaatit ja koon
- Käyttäjä ei voi muuttaa ikkunan kokoa
- Fonttien muutos voi katkaista tekstin
- Kaikki tyylit eivät välttämättä toimi (komponenttien kokojen takia)
- Ohjelman ylläpitäminen työlästä

Manuaalinen layout

- Ohjelmoija määrittää komponenttien paikat, mutta niiden koot lasketaan ikkunan koon mukaan.
- Tehdään ylikirjoittamalla dialogin resizeEvent() –funktio
- Lopputulos usein parempi kuin absoluuttisilla koordinaateilla, mutta toteuttaminen työlästä

Qt:n layout-managerit

Yksinkertainen tapa käyttöliittymäkomponenttien automaattiseen järjestämiseen

- kuvaavat, miten komponentit asettuvat käyttöliittymään
- automaattisesti asemoivat komponentteja ja säätävät niiden kokoa
- takaavat, että komponenteilla on säännönmukainen järjestys ja että käyttöliittymä pysyy käytettävänä

Qt:n layout-managerit

- Kokomääritykset summittaisia
  - Minimum size
  - Maximum size
  - Preferred size
- Widgetti voi olla skaalautuva tai säilyttää oman oletuskoon (preferred size)
- Layout-manageri laskee widgetin koon ja sijainnin uudelleen aina, kun layout muuttuu tai ikkunan kokoa muutetaan – Resize event

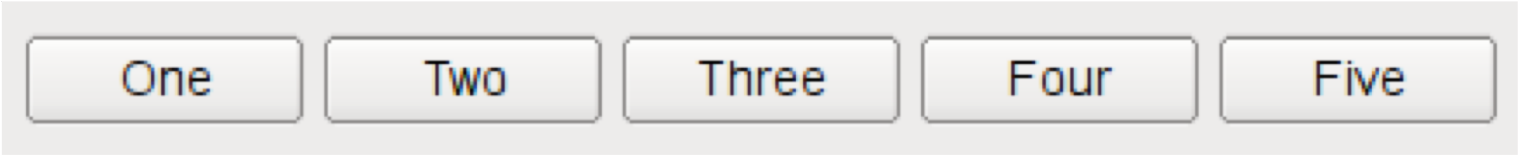
Qt:n layout-managerit

- Suoraviivaisin tie nättiin asemointiin on Qt:n valmiit layout-managerit:
  - `QHBoxLayout`
  - `QVBoxLayout`
  - `QGridLayout`, ja
  - `QFormLayout`

    joita on mahdollista latoa sisäkkäin

QHBoxLayout

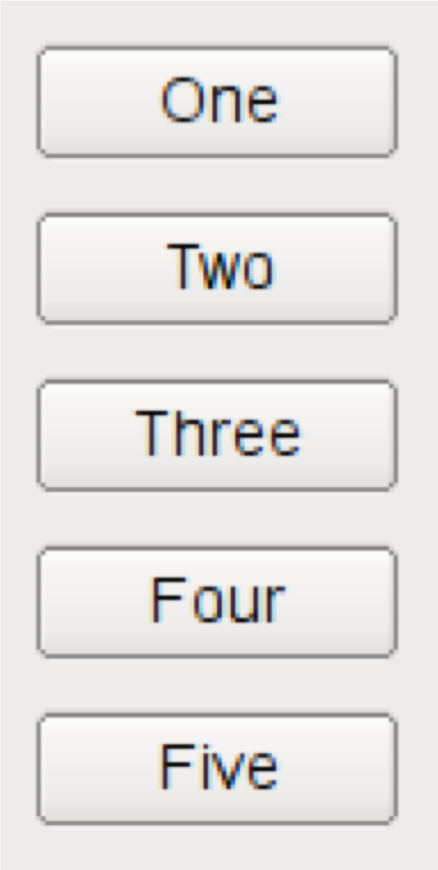
Latoo komponentit vaakasuoraan



Kuva: Qt:n dokumentaatio

QVBoxLayout

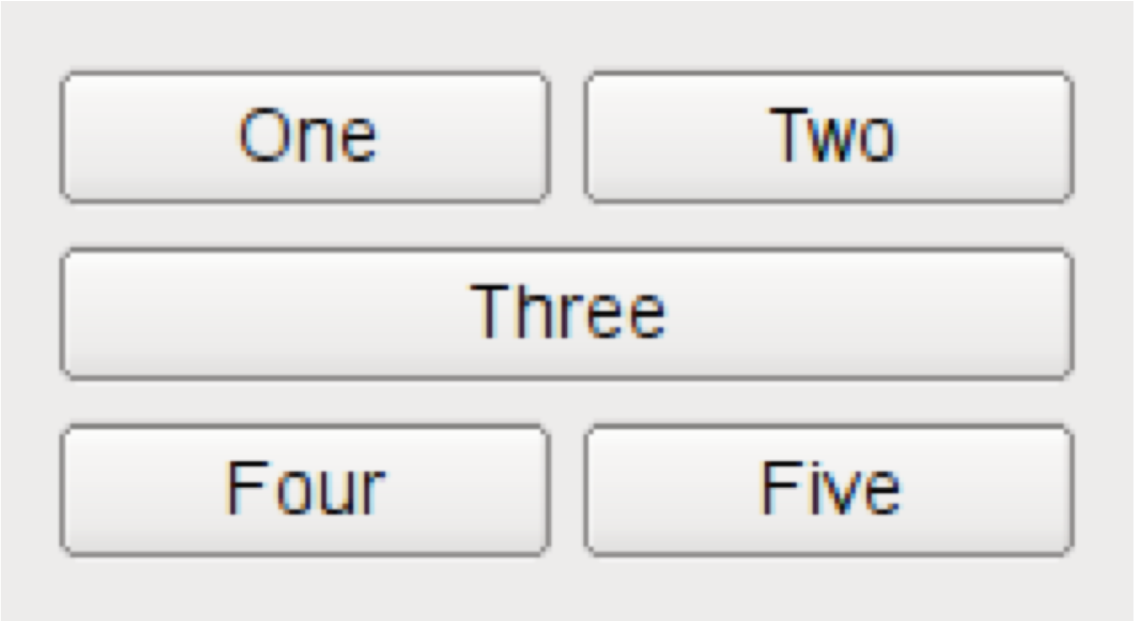
Latoo komponentit pystysuoraan



Kuva: Qt:n dokumentaatio

QGridLayout

Latoo komponentit ruudukkoon



Kuva: Qt:n dokumentaatio

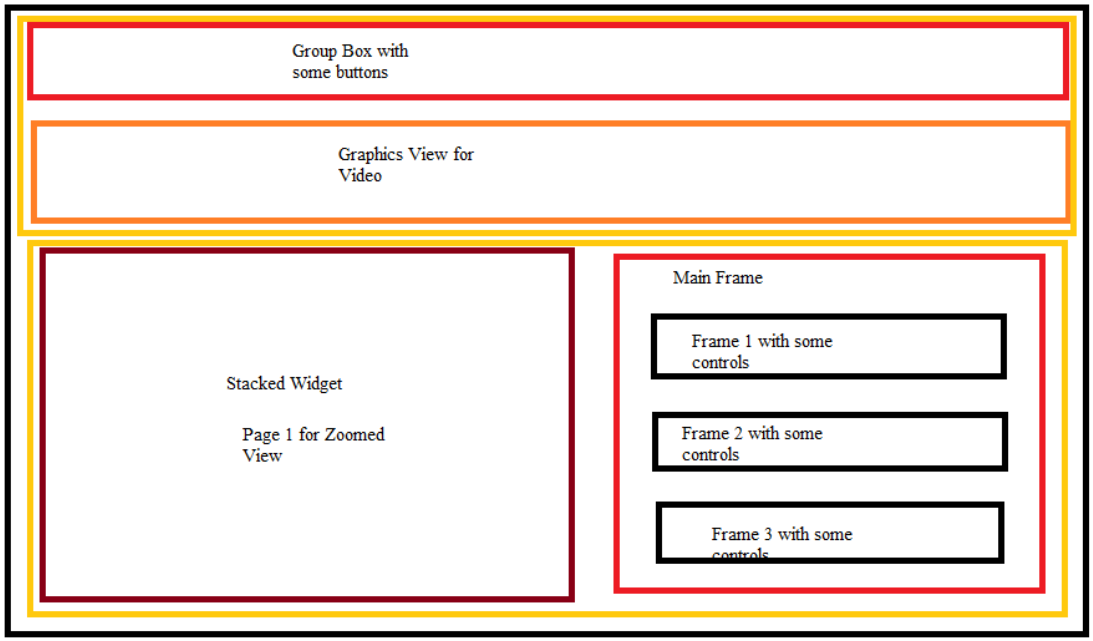
QFormLayout

Latoo komponentit kahteen sarakkeeseen

Kuva: Qt:n dokumentaatio

Sisäkkäiset layoutit

Dialogi-ikkuna koostuu tyypillisesti useista sisäkkäisistä layouteista



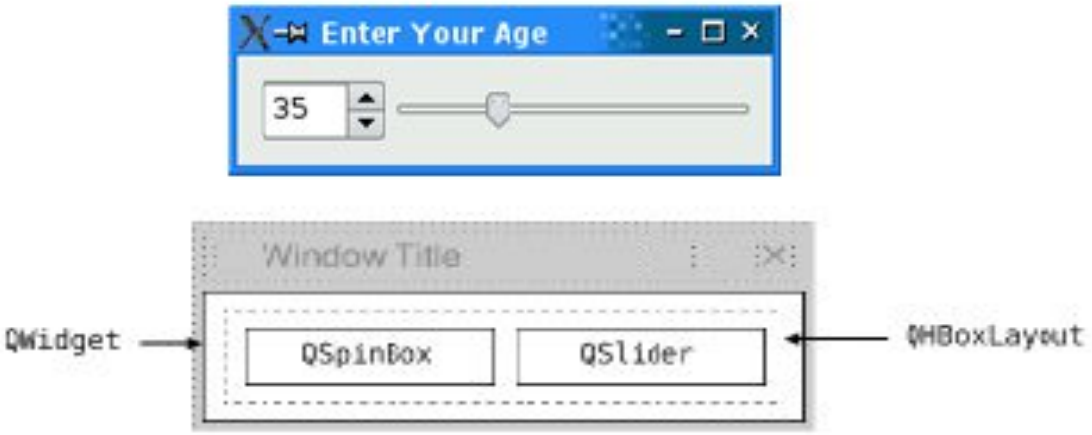
Kuva: Stackoverflow

Parent

- Layout asettaa widgeteille automaattisesti parent-widgetin (reparent)
- Layoutiin asetetun widgetin parent-widget ei ole layout itse vaan widgetti, johon layout on asetettu (setLayout)

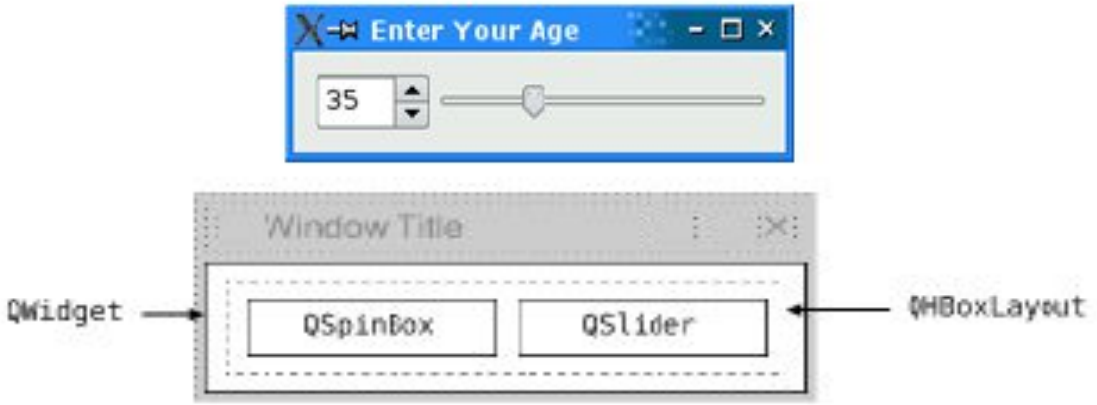
Esimerkki

- Kolme widgettia:
  - parent: QWidget (pääikkuna)
  - children: QSpinBox ja QSlider
- Signaalit ja slotit synkronoivat widgetteja

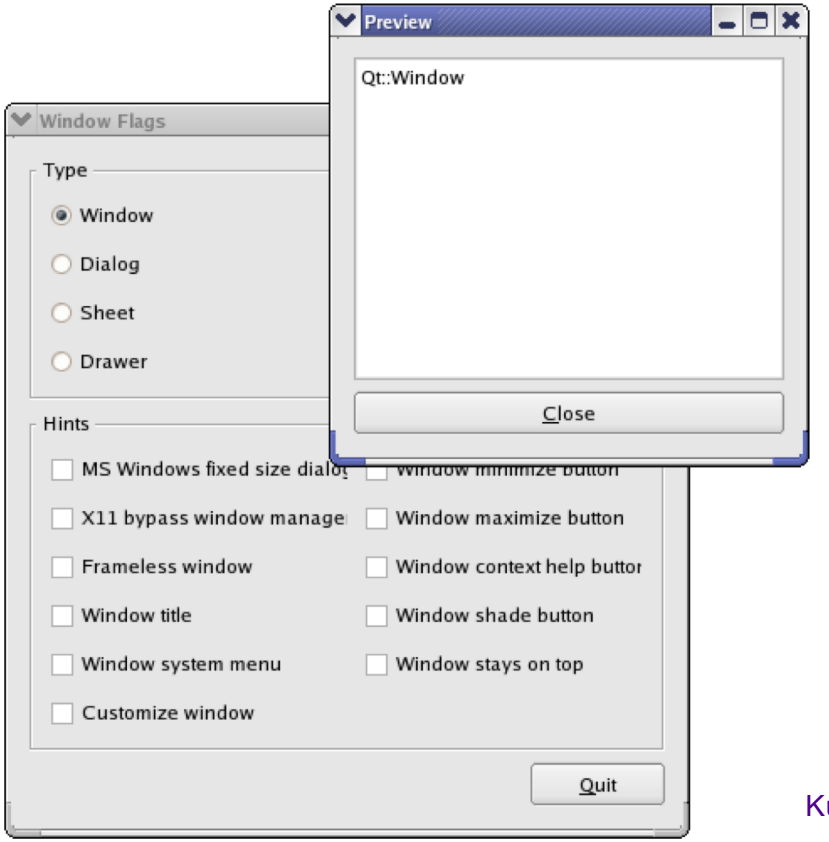


```
#include <QApplication>
#include <QHBoxLayout>
#include <QSlider>
#include <QSpinBox>

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);          // jatkuu
```

```
    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();
}
```



Usean ikkunan käytöstä



Kuva: Qt:n dokumentaatio