|  |  |
| --- | --- |
| Декларативна мова програмування |  |

Ø **QML** (Qt Meta Language або Qt Modeling Language) – декларативна мова програмування :

* заснована на JavaScript;
* призначена для дизайну додатків, що роблять основний упор на інтерфейс користувача;
* є частиною Qt Quick, середовища розробки інтерфейсу користувача, що поширюється разом з Qt ;
* в основному використовується для створення програм, орієнтованих на мобільні пристрої з сенсорним керуванням.

Ø **Переваги**

• Код пишеться один раз і може використовуватись на кількох платформах;

• Розробка компактних, високопродуктивних програм;

Ø **Базова архітектура**

· QML -документ є дерево елементів. QML елемент, як і елемент Qt , є сукупність блоків: графічних (таких, як rectangle, image) і поведінкових (таких, як state, transition, animation). Ці елементи можуть бути об'єднані, щоб побудувати комплексні компоненти, починаючи від простих кнопок та повзунків і закінчуючи повноцінними програмами, що працюють з Інтернетом.

· *Сигнали та обробники сигналів (Signals & Signal Handler)* :

* QML має механізм сигналу та обробника, де *signal* є подією, і на сигнал реагує через *сигнал handler* ;
* *конкретний сигнал викликається для конкретного об'єкта, потрібно в об'єкті оголосити обробник сигналу, названий on <Signal>, де <Signal> - ім'я* сигналу , *з* першою *великою літерою* .

Наприклад :

Button {  
 anchors.bottom: parent.bottom  
 anchors.horizontalCenter: parent.horizontalCenter  
 text: "Change color!"  
 onClicked: {  
 rect.color = Qt.rgba(Math.random(), Math.random(), Math.random(), 1);  
 }  
 }

* У деяких випадках бажано отримати доступ до сигналу поза об'єктом, що його викликає. Для цього в QtQuick передбачено тип [Connections](https://doc.qt.io/qt-5/qml-qtqml-connections.html) для підключення до сигналів довільних об'єктів.
* Button {  
   id: button  
   anchors.bottom: parent.bottom  
   anchors.horizontalCenter: parent.horizontalCenter  
   text: "Change color!"  
   }  
    
   Connections {  
   target: button  
   onClicked: {  
   rect.color = Qt.rgba(Math.random(), Math.random(), Math.random(), 1);  
   }  
   }

**Вправи для самостійного опрацювання:**

1. а) Створити додаток, який виводить "Hello world" в середині головного вікна.

Шрифт:

- Розмір у поінтах: 13;

- Arial;

- курсив;

- колір червоний.

б) Після натискання на область поза текстом, текст повинен помінятися на "Clicked" і колір змінити на зелений.

2. містить три прямокутники різного кольору в одному рядку.

* перший має ширину 100 пікселів;
* останній залежить від ширини першого (наприклад, його ширина = 2\* ширини першого);
* у другого ширина, що залишилася. При цьому **другому розраховувати ширину не можна;**
* кожен із прямокутників має відступ з кожної сторони.

3. Написати додаток Qt C ++, яке створює MyModel всередині якої std::vector<MyStruct>. MyStruct містить два поля (наприклад, рядок та число). Заповніть у конструкторі вектор тестовими даними.

main.qml відображає дані з MyModel використовуючи ListView . Для доступу до кожного з полів MyStruct використовувати ролі (одна роль повертає одне поле). Після отримання кожного з полів їх сконкатенувати та вивести на екран.