Porting to QML - The Cautious Approach

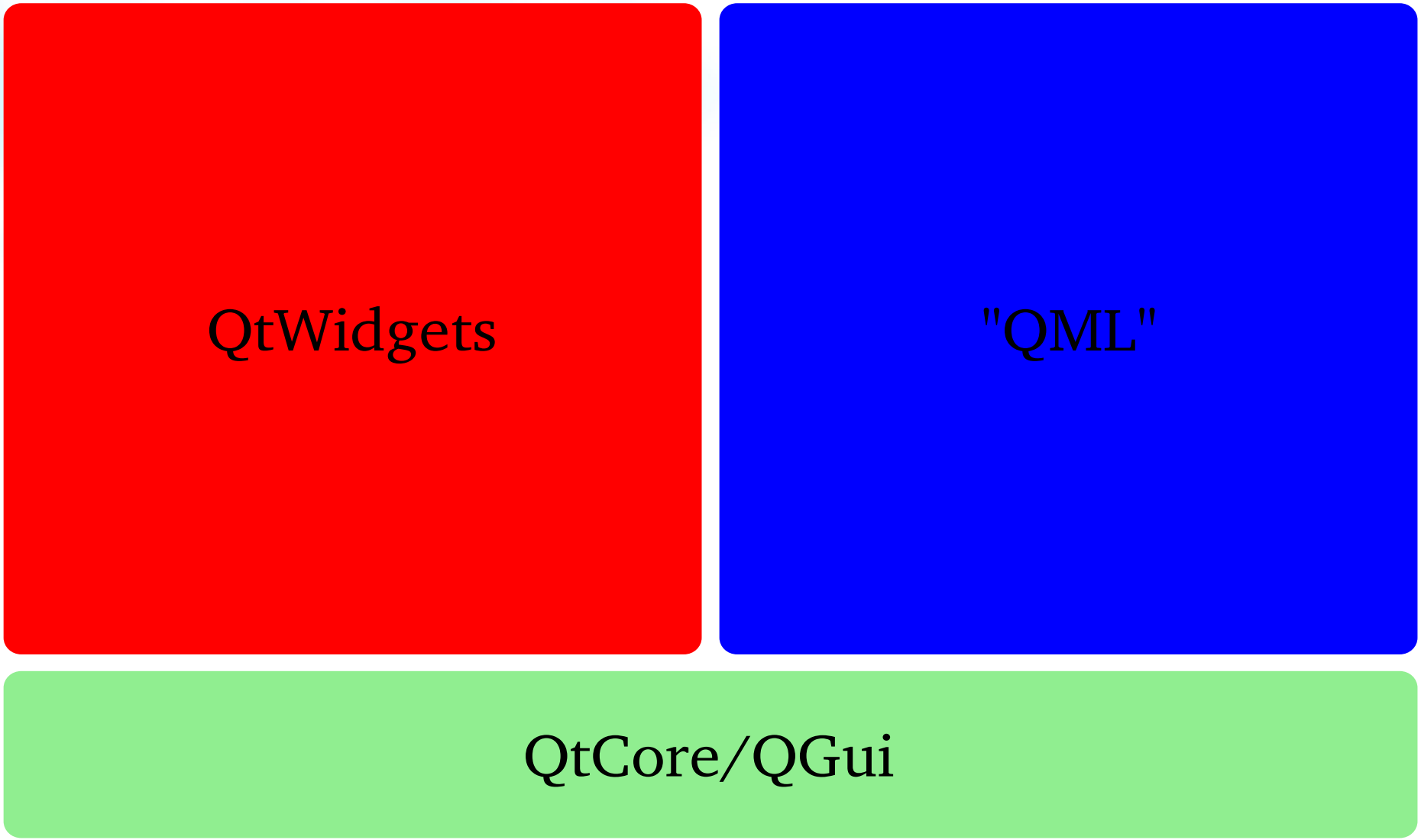
Akademy 2014

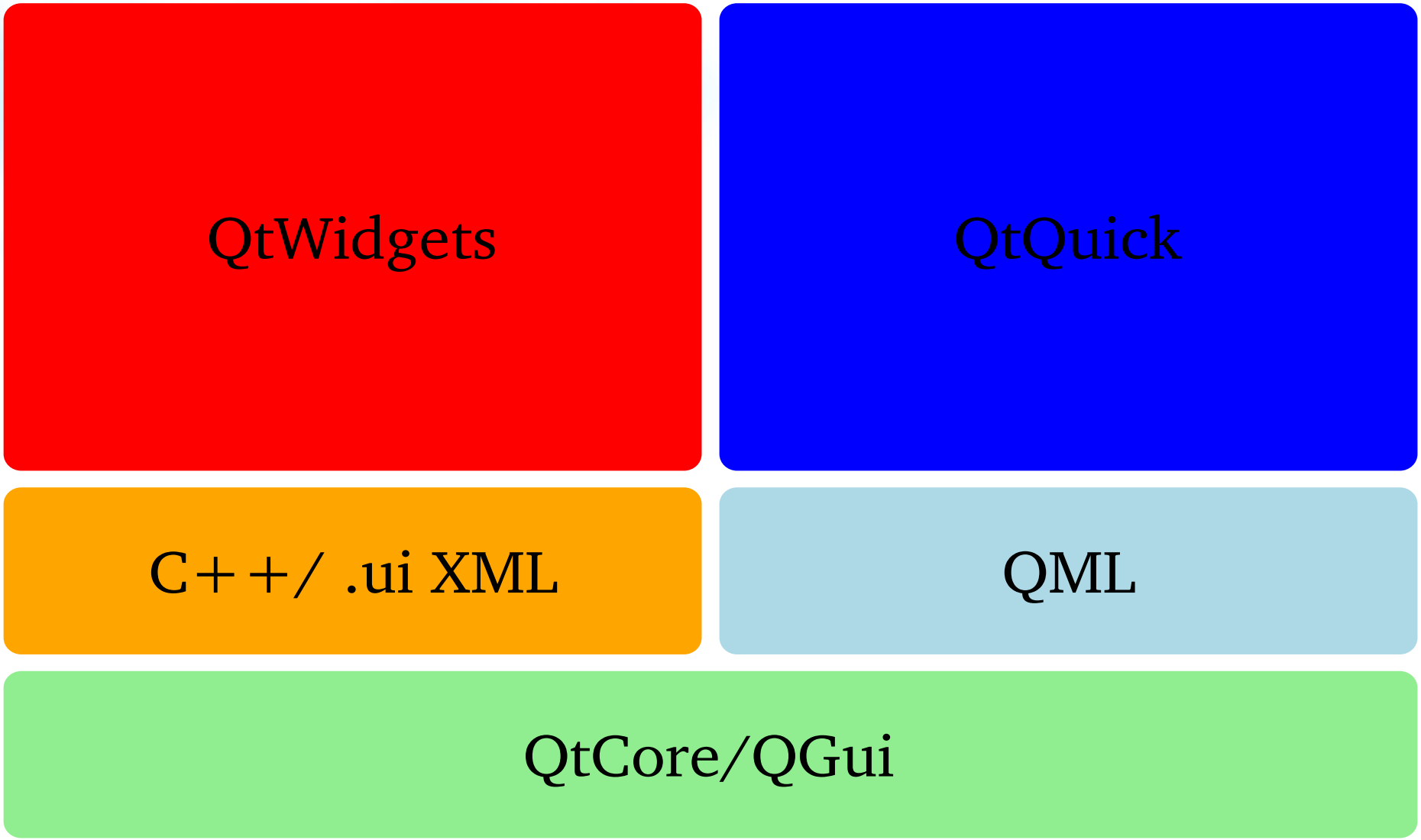
Presented by Kevin Krammer

krammer@kde.org

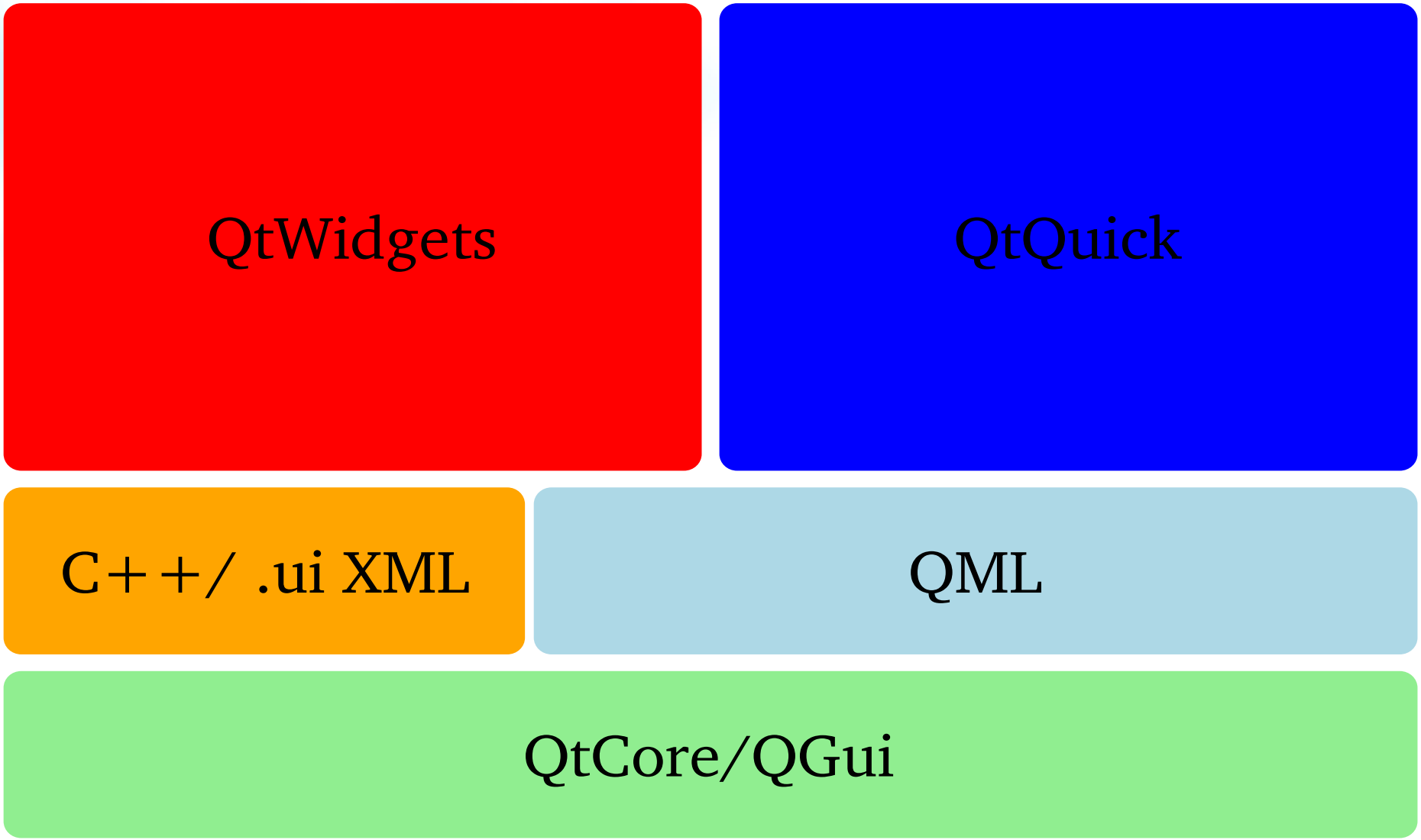
- **The stack misconception**
- C++ and QML: Concepts and Differences
- Demo
- Conclusions

Common perception Widgets vs. QML




QtWidgets
QtQuick
C++/ .ui XML
QML
QtCore/QGui

But this is also possible



- QML describes a tree of objects
- Can instantiate any QObject subclass that has a default constructor
- QWidget is derived from QObject
- Most widget classes do not require constructor arguments

For example KDAB's DeclarativeWidgets:
https://github.com/KDAB/DeclarativeWidgets

- The stack misconception
- **C++ and QML: Concepts and Differences**
- Demo
- Conclusions

```
1 class Contact
2 {
3 public:
4     void setFirstName(const QString &firstName);
5     QString firstName() const;
6
7 private:
8     QString mFirstName;
9 };
```

- C++
  - create instances
  - copy instances
  - direct calls to all methods
- QML
  - move around as QVariant
  - no access to methods

Solution: QObject based adapters/mediators

Widget Values: Push/Pull vs. Observers

- C++
  - Call to widget setters (push data)
  - Call to widget getters (pull data)
- QML
  - Widgets observe objects' properties

```
ui->firstName->setText(contact.firstName());
...
contact.setFirstName(ui->firstName()->text());
```

```
ContactEditor {
    id: contactEditor
    firstName: firstNameInput.text
}

LineEdit {
    id: firstNameInput
    text: contactEditor.firstName
}
```

Data Changes: React to change vs. React to new value

- C++
  - Changes result in signals, react in connected slots
- QML
  - Mostly bind one object's properties to other objects
  - Value gets reevaluated when parts of the expressions change value

```
// C++
void SomeClass::onInputChanged()
{
    const bool acceptableInput = !ui->firstName->text().isEmpty() ...
    ui->saveButton->setEnabled(acceptableInput);
}
```

```
// QML
PushButton {
    text: "Save"

    enabled: firstName.text.length() > 0 ...
}
```

- The stack misconception
- C++ and QML: Concepts and Differences
- **Demo**
- Conclusions

A simple Contact Editor

- Step 0: Standard QtWidget application
- Step 1: Move data handling into non-UI interface object
- Step 2: Prepare interface objects for QML
- Step 3: Port to QML

- The stack misconception
- C++ and QML: Concepts and Differences
- Demo
- **Conclusions**

- Porting can be done in several steps
- Each step keeping the current application fully functional
- Widgets UI can remain an option parallel to QtQuick

Resources

- DeclarativeWidgets: https://github.com/KDAB/DeclarativeWidgets
- Talk at Akademy 2013: https://conf.kde.org/en/Akademy2013/public/events/20
- Talk at Qt Developer Days 2013: https://www.youtube.com/watch?v=NqpJEj15t9Q
- Talk me:
  - Here at Akademy until Friday
  - email: krammer@kde.org
  - IRC: krake on #akademy, #kde-devel, #kontact