How I integrated TinyMCE into my Qt Application

Jens Weller

CppCon 2015 lightning talks

About me

- C++ Evangelist
  - @meetingcpp

- C++ since '98
- '02-'07 Vodafone
- '07 selfemployed / freelancer in C++
- '12 Meeting C++

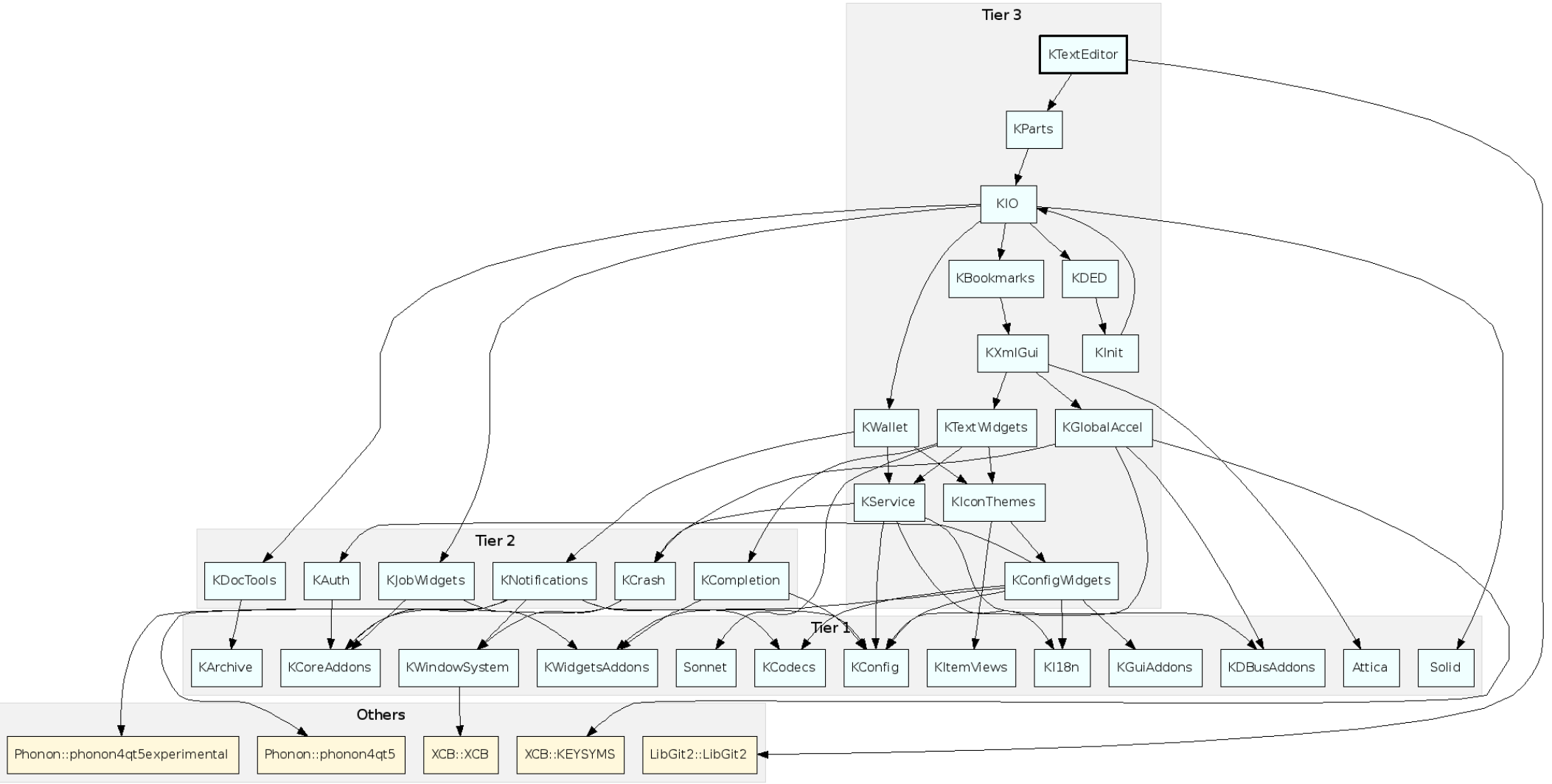
TextEditor needed

- Qt offers TextControls
  - Editing and displaying simple text
  - Can also display HTML and Richtext
    - Very basic editing support
- KDE Frameworks 5
  - Has a KTextEditor Class
  - Probably the best C++ Solution for Qt
- wxWidgets has also a library solution

KDE Frameworks 5

- Has KTextEditor
- Seems the best option in C++ UI Land
  - For Qt
  - wxWidgets has afaik some library...
- KDE
  - Windows Support?
    - Should work with Qt
  - Alternatives?

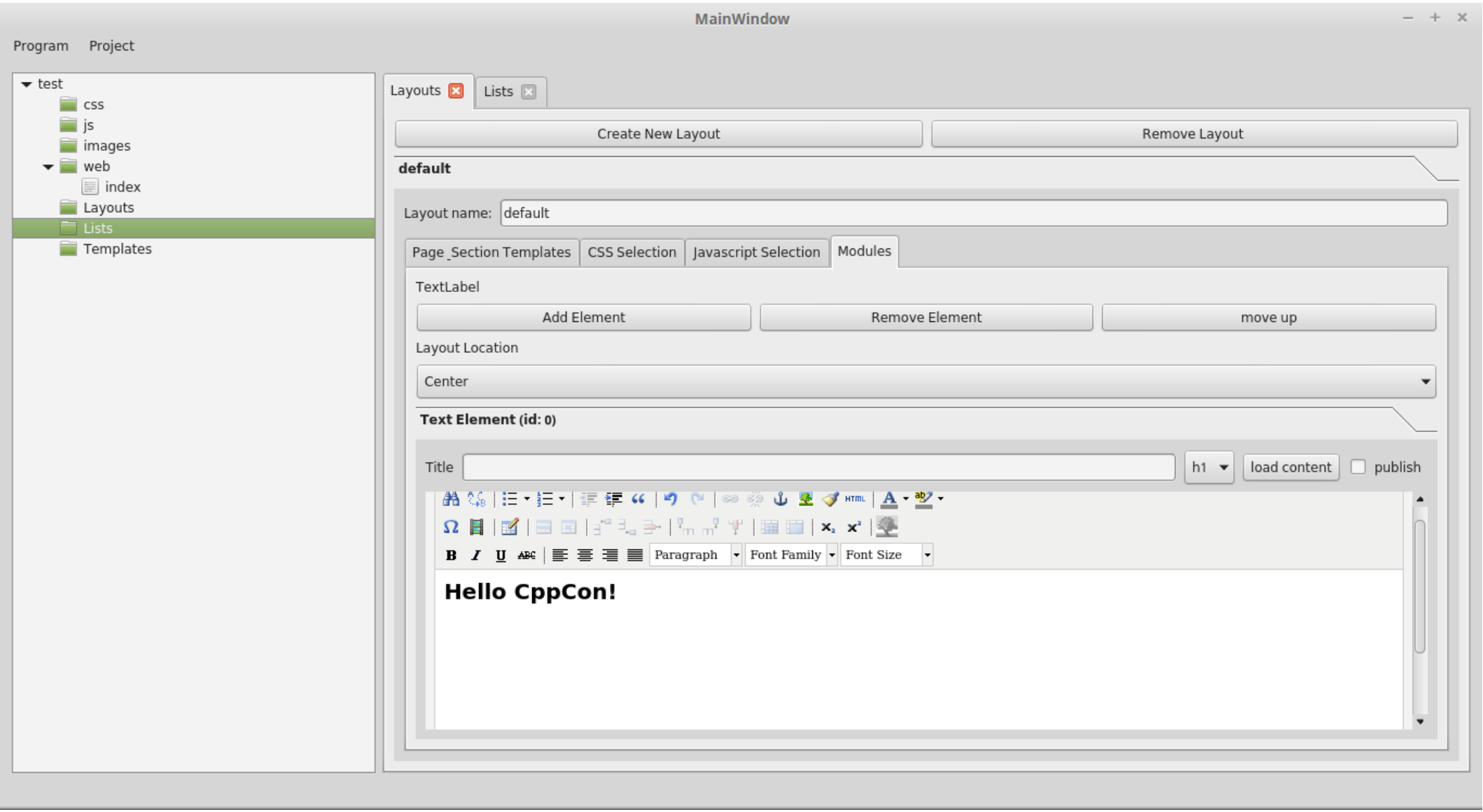
KTextEditor dependencies



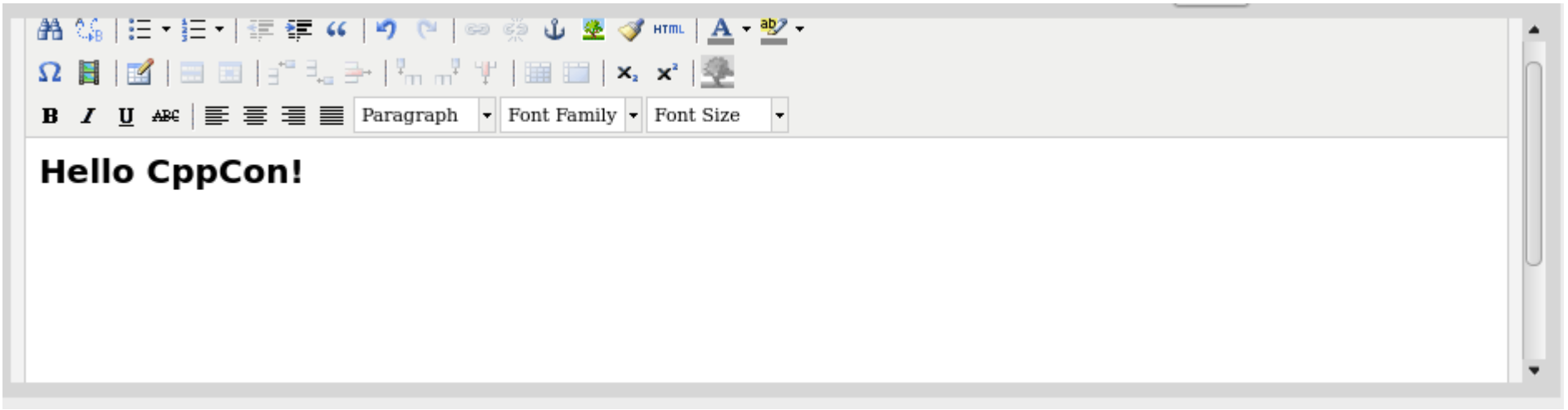
What I need

- HTML Wysiwyg Editor
- Display and edit HTML Parts
  - Links
  - Images
- TinyMCE
  - Is what is currently used
  - Has all needed features

Endresult


MainWindow
Program
Project
test
css
js
images
web
index
Layouts
Lists
Templates
Layouts
Lists
Create New Layout
Remove Layout
default
Layout name:
default
Page _Section Templates
CSS Selection
Javascript Selection
Modules
TextLabel
Add Element
Remove Element
move up
Layout Location
Center
Text Element (id: 0)
Title
h1
load content
publish
Paragraph
Font Family
Font Size
Hello CppCon!

Endresult



How to get there?

- TinyMCE
  - JavaScript
  - HTML
  - “Browser technology”
- Qt Webview
  - Webkit based
  - C++/JS Bridge
  - Renders HTML
  - C++ API

TinyMCE Limitations

- What is in the browser, stays there
  - Dialogs can't leave browser window
  - Some Dialogs are too large
- This is a hack
  - Official interfaces are not meant to run in an application context
  - image_list.js
    - Returns list of images
      - var ImageList ={{“foo”,”foo.jpg”}{“bar”,”bar.png”}};
    - Is a js file or (php) script returning a js file
      - Executable?
    - http://127.0.0.1/image_list.js Does not work!

So ...

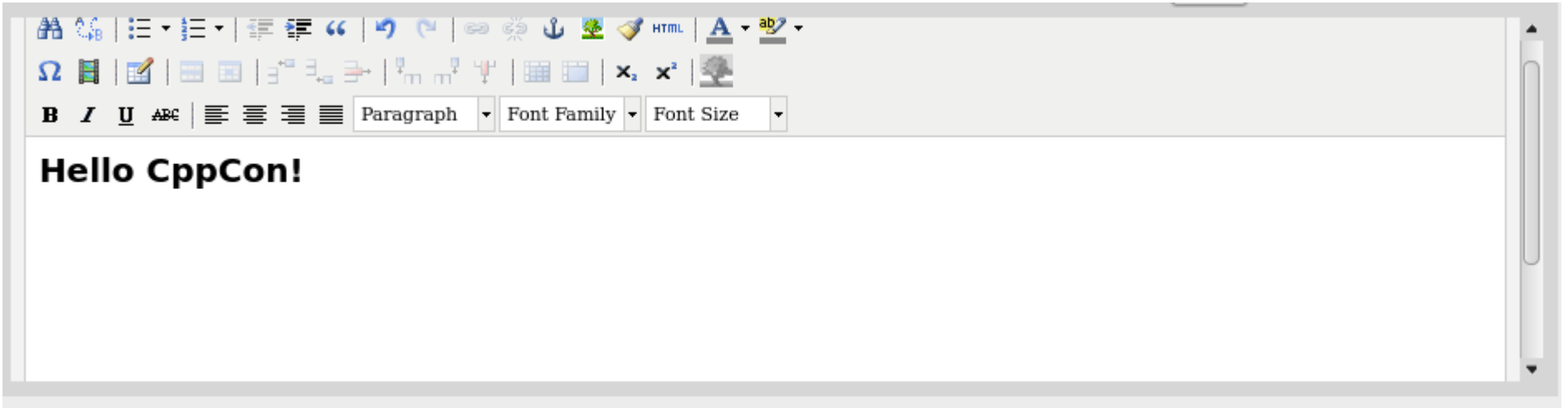
- TinyMCE “Features”
  - JS Dialogs are not very useful
    - → replace with Qt Dialog?
  - Integration of Images/Links difficult
    - → replace official integration with something?

QWebkit

- QWebkit JS/C++ Bridge
  - Register QObjects into JavaScript
  - Call methods marked with Q_INVOKABLE
  - Emit Signals from JavaScript
  - Execute JavaScript from C++ context
- Roundtrip
  - JS → C++ → JS

Class HTMLTextEditor

- Derived from QWebView
- C++ Interface for the Text Editor



Class HTMLTextEditor

```
class HTMLTextEditor : public QWebView
{
    Q_OBJECT
public:
    explicit HTMLTextEditor(QWidget *parent = 0);

    void setImagelist(const QStringList &value);
    void setLinklist(const QHash<size_t,QString> &value);
    Q_INVOKABLE void insertLink();
    QString text()const{return getContent();}

signals:
    void selectImage();
private slots:
    void onSelectImage();
    void initEditor();

private:
    QVariant execJS(const QString& js) const;
};
```

Class HTMLTextEditor

```
class HTMLTextEditor : public QWebView
{
    Q_OBJECT
public:
    explicit HTMLTextEditor(QWidget *parent = 0);

    void setImagelist(const QStringList &value);
    void setLinklist(const QHash<size_t,QString> &value);
    Q_INVOKABLE void insertLink();
    QString text()const{return getContent();}

signals:
    void selectImage();
private slots:
    void onSelectImage();
    void initEditor();

private:
    QVariant execJS(const QString& js) const;
};
```

Class HTMLTextEditor

```
class HTMLTextEditor : public QWebView
{
    Q_OBJECT
public:
    explicit HTMLTextEditor(QWidget *parent = 0);

    void setImagelist(const QStringList &value);
    void setLinklist(const QHash<size_t,QString> &value);
    Q_INVOKABLE void insertLink();
    QString text()const{return getContent();}

signals:
    void selectImage();
private slots:
    void onSelectImage();
    void initEditor();

private:
    QVariant execJS(const QString& js) const;
};
```

Class HTMLTextEditor

```
class HTMLTextEditor : public QWebView
{
    Q_OBJECT
public:
    explicit HTMLTextEditor(QWidget *parent = 0);

    void setImagelist(const QStringList &value);
    void setLinklist(const QHash<size_t,QString> &value);
    Q_INVOKABLE void insertLink();
    QString text()const{return getContent();}

signals:
    void selectImage();
private slots:
    void onSelectImage();
    void initEditor();

private:
    QVariant execJS(const QString& js) const;
};
```

Class HTMLTextEditor

```
class HTMLTextEditor : public QWebView
{
    Q_OBJECT
public:
    explicit HTMLTextEditor(QWidget *parent = 0);

    void setImagelist(const QStringList &value);
    void setLinklist(const QHash<size_t,QString> &value);
    Q_INVOKABLE void insertLink();
    QString text()const{return getContent();}

signals:
    void selectImage();
private slots:
    void onSelectImage();
    void initEditor();

private:
    QVariant execJS(const QString& js) const;
};
```

Implementation

- C++
  - Expose class to JS
  - Image/Link Handlers
    - Display Qt Dialog
    - Execute JS
- TinyMCE
  - Replace dialogs
    - Write plugin
    - Call into C++
- editor.html
  - HTML Host file
  - Contains
    - TextArea
    - JS to load TinyMCE
  - Location is Base Path

C++ Implementation

- Constructor
  - Prevent Links from Opening inside QWebView

```
HTMLTextEditor::HTMLTextEditor(QWidget *parent) :
    QWebView(parent)
{
    page()->setLinkDelegationPolicy(QWebPage::DelegateExternalLinks);
    connect(this,SIGNAL(selectImage()),this,SLOT(onSelectImage()));
}
```

C++ Implementation

- setBasePath
  - Basic setup for the editor

```
void HTMLTextEditor::setBasePath(const QString &bp)
{
    basepath = bp;
    setUrl(QUrl("file:///"+basepath+"/editor.html"));
    mainframe = page()->mainFrame();
    mainframe->addToJavaScriptWindowObject("hostObject",this);
    //QTimer::singleShot(200,this,SLOT(initEditor()));
}
```

C++ Implementation

- setBasePath
  - Basic setup for the editor

```
void HTMLTextEditor::setBasePath(const QString &bp)
{
    basepath = bp;
    setUrl(QUrl("file:///"+basepath+"/editor.html"));
    mainframe = page()->mainFrame();
    mainframe->addToJavaScriptWindowObject("hostObject",this);
    //QTimer::singleShot(200,this,SLOT(initEditor()));
}
```

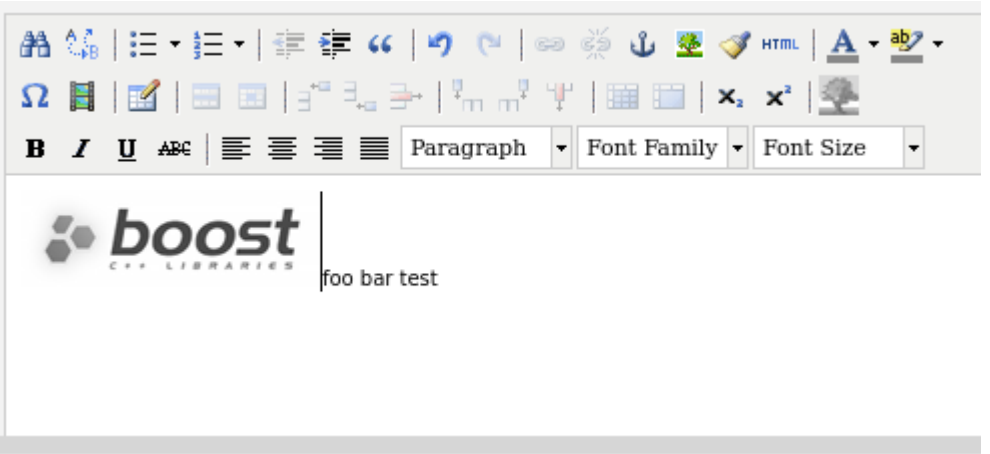
C++ Implementation

- onSelectImage
  - Select image and set HTML in the editor

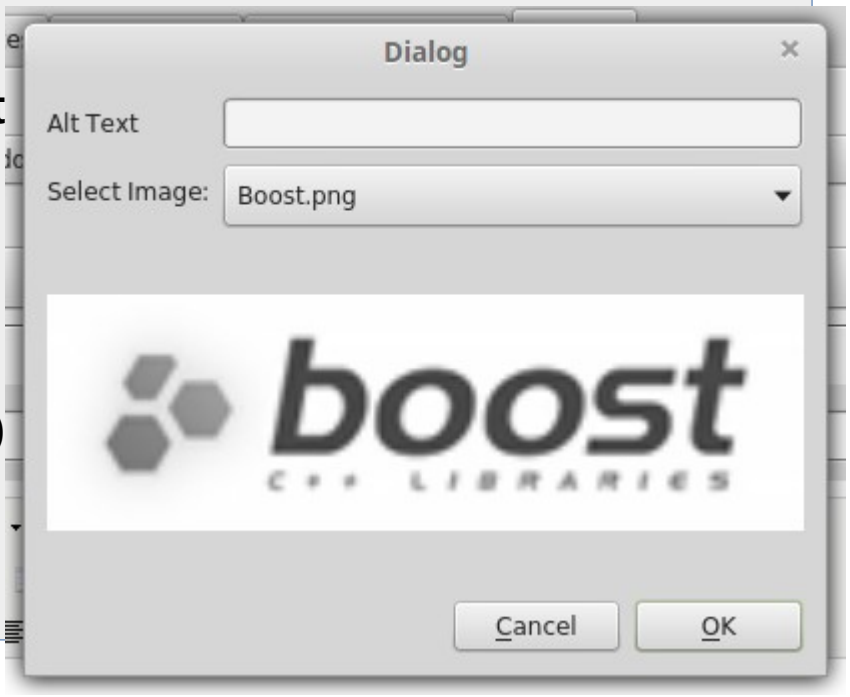
```
void HTMLTextEditor::onSelectImage()
{
    ImageDialog dlg(basepath + "/img/",imagelist,this);
    if(dlg.exec()!= QDialog::Accepted)return;
    QString alt,img;
    dlg.transferData(alt,img);
    QString js = R"(ed = tinyMCE.activeEditor;
ed.execCommand('mceInsertContent',false,
ed.dom.createHTML('img',{src : "img/%1",alt : "%2"}), {skip_undo : 1});
ed.undoManager.add();)";
    execJS(js.arg(relative + img,alt));
}
```

C++ Implementation

- onSelectImage
  - Select image and set




```
void HTMLTextEditor::onSelectImage()
{
    ImageDialog dlg(basepath + "/img/",imagelist,t
    if(dlg.exec()!= QDialog::Accepted)return;
    QString alt,img;
    dlg.transferData(alt,img);
    QString js = R"(ed = tinyMCE.activeEditor;
ed.execCommand('mceInsertContent',false,
ed.dom.createHTML('img',{src : "img/%1",alt : "%2"})
ed.undoManager.add();)";
    execJS(js.arg(relative + img,alt));
}
```



C++ Implementation

- onSelectImage
  - Select image and set HTML in the editor

```
void HTMLTextEditor::onSelectImage()
{
    ImageDialog dlg(basepath + "/img/",imagelist,this);
    if(dlg.exec()!= QDialog::Accepted)return;
    QString alt,img;
    dlg.transferData(alt,img);
    QString js = R"(ed = tinyMCE.activeEditor;
ed.execCommand('mceInsertContent',false,
ed.dom.createHTML('img',{src : "img/%1",alt : "%2"}), {skip_undo : 1});
ed.undoManager.add();)";
    execJS(js.arg(relative + img,alt));
}
```

C++ Implementation

- onSelectImage
  - Select image and set HTML in the editor

```
void HTMLTextEditor::onSelectImage()
{
    ImageDialog dlg(basepath + "/img/",imagelist,this);
    if(dlg.exec()!= QDialog::Accepted)return;
    QString alt,img;
    dlg.transferData(alt,img);
    QString js = R"(ed = tinyMCE.activeEditor;
ed.execCommand('mceInsertContent',false,
ed.dom.createHTML('img',{src : "img/%1",alt : "%2"}), {skip_undo : 1});
ed.undoManager.add();)";
    execJS(js.arg(relative + img,alt));
}
```

C++ Implementation

- execJS
  - Log and execute JavaScript

```
QVariant HTMLTextEditor::execJS(const QString &js)const
{
    qDebug() << "exec js" << js;
    return mainframe->evaluateJavaScript(js);
}
```

JS Implementation

- I'm not very good at JavaScript
  - Actually, that is the first time, I really need to write and understand JS code...

```
ed.addCommand('mceQtImage', function() {
    window.hostObject.selectImage();
                });
```

```
ed.addButton('qtimage', {
                    title : 'qtimage.desc',
                    cmd : 'mceQtImage',
                    image : url + '/img/image.gif'
                });
```

Current limitations

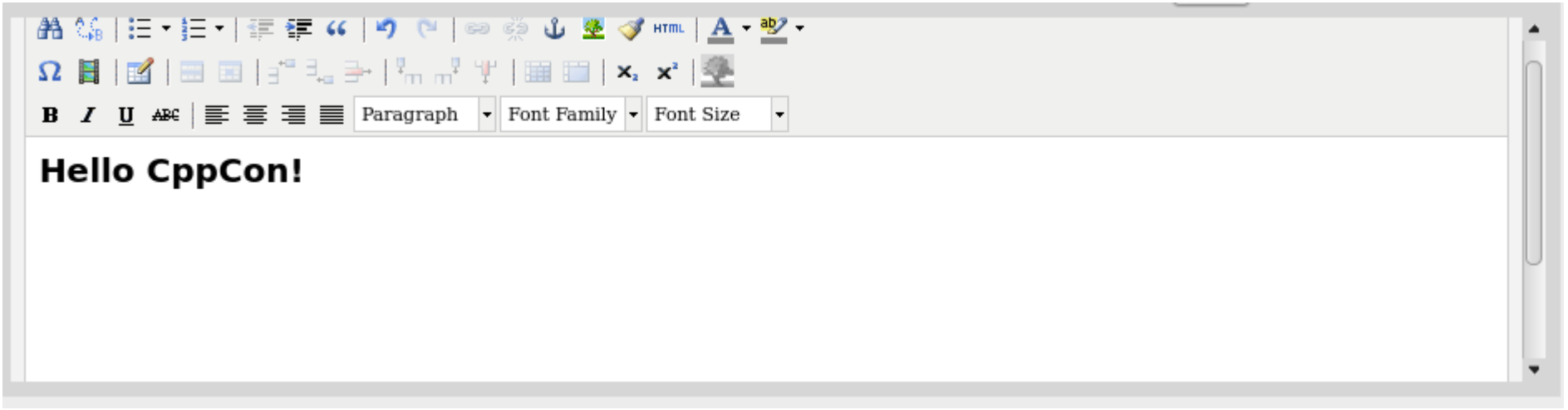
- Lots of resources get allocated for this
  - One Editor would be enough
  - Instead we have n editors loaded
- Basepath and other limitations
  - One editor.html per directory currently
    - Workaround
  - TinyMCE.baseURI.setPath does not work
- JavaScript and WebKit is difficult to debug

What I also would need...

- A Text Editor for non HTML Text
  - CSS
  - JavaScript
- TinyMCE isn't really good for this
- Maybe other Webeditors would do
- Atom
  - Runs in NodeJS, not really an alternative

Endresult

- Working HTML Editor
- Some fine tuning still needed
- Some parts will always be a hack
- TinyMCE 4.x didn't run in QWebView



Thanks for listening!

Questions?

Jens Weller

@meetingcpp
info@meetingcpp.com