QT WEBCHANNEL

BRIDGING THE GAP BETWEEN HTML AND QT

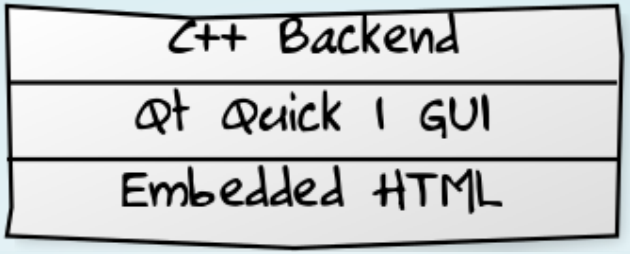
Milian Wolff / www.kdab.com

OUTLINE

- Motivation
- Qt WebChannel
- Outlook

MOTIVATION

QT WEBKIT 1



Qt WebKit Bridge

PUBLISH A C++ QOBJECT

```
QWebFrame *frame = myWebPage->mainFrame();
frame->addToJavaScriptWindowObject("foo", myObject);
/* ... */
```

PUBLISH A QT QUICK 1 OBJECT

```
import QtWebKit 1.0

... {
    QtObject {
        id: myObject,
        WebView.windowObjectName: "foo"
        // more properties, signals, functions...
    }
    WebView {
      javaScriptWindowObjects: [myObject /*, ...*/]
    }
}
```

ACCESSING AN OBJECT FROM JAVASCRIPT

```
foo.someSignal.connect(function(arg1 /*, arg2, ...*/) { /*...*/ });

foo.someProperty = foo.someOtherProperty + 1;

var ret = foo.someMethod(1, "asdf", [1, 2], {asdf: "bar"});
```

QT WEBKIT BRIDGE

- easy to use
- efficient
- synchronous API
- only for Qt WebKit 1
- no support for Qt Quick 2

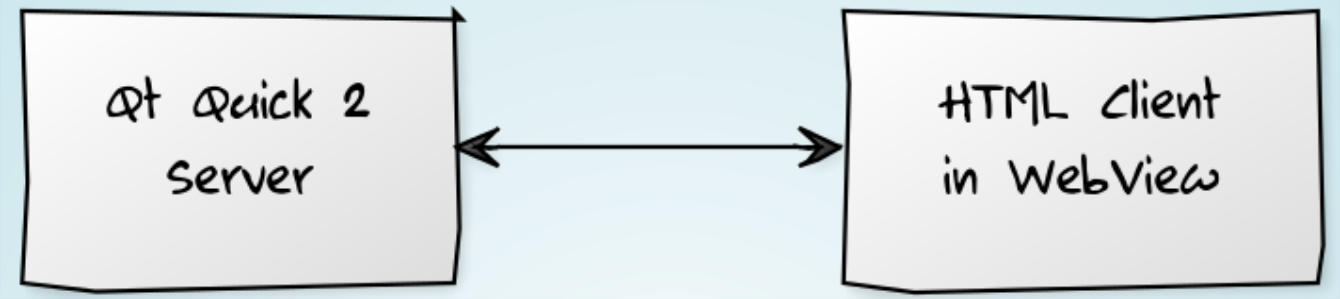
QT WEBKIT 2 & BEYOND

QT WEBCHANNEL

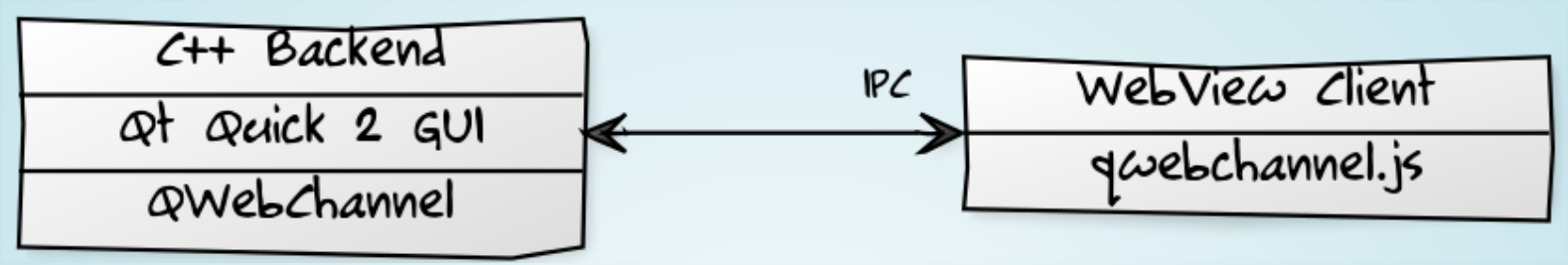
HOW?

- serialize Qt MetaObject
- send to HTML process via IPC
- create JavaScript object with mirrored API
- transmit signals, method calls, property changes

DEMO



SETUP


C++ Backend
Qt Quick 2 GUI
QWebChannel
IPC
WebView Client
qwebchannel.js

PUBLISH A C++ QOBJECT

```
QWebChannel channel;
channel.registerObject(QStringLiteral("foo"), foo);
/* ... */
channel.connectTo(someClient);
```

PUBLISH A QT QUICK 2 OBJECT

```
import QtWebChannel 1.0
import QtWebKit 3.0
import QtWebKit.experimental 1.0

QtObject {
    id: foo,
    WebChannel.id: "foo"
    // more properties, signals, functions...
}
WebView {
    experimental.webChannel.registeredObjects: [
        foo /*, ... */
    ];
}
```

ACCESSING AN OBJECT FROM JAVASCRIPT

```
<script type="text/javascript"
        src="qrc:///qtwebchannel/qwebchannel.js"></script>
```

```
// in your window.onload handler or similar:
new QWebChannel(navigator.qtWebChannelTransport,
                function(channel) {
    // connection to server succeeded, objects available via:
    var foo = channel.objects.foo;
    // use them like before:
    foo.someSignal.connect(function(arg1 /*, arg2, ...*/) {
        /*...*/
    });
    foo.someProperty = foo.someOtherProperty + 1;
    // big difference: async return values
    foo.someMethod(1, "asdf", [1, 2], {asdf: "bar"},
                   function(ret) { /* method call returned */ });
});
```

QT WEBCHANNEL TRANSPORTS

```
class QWebChannelAbstractTransport : public QObject
{
    Q_OBJECT
public:
    explicit QWebChannelAbstractTransport(QObject *parent = 0);
    virtual ~QWebChannelAbstractTransport();

public Q_SLOTS:
    virtual void sendMessage(const QJsonObject &message) = 0;

Q_SIGNALS:
    void messageReceived(const QJsonObject &message,
                         QWebChannelAbstractTransport *transport);
};
```

existing implementations:

- Qt WebKit 2 IPC
- Qt WebEngine IPC
- Qt WebSockets

SIDE-EFFECT

works in any JavaScript runtime with WebSockets:

- desktop browsers
- mobile browsers (e.g. via Qt WebView)
- node.js
- Qt Quick 2

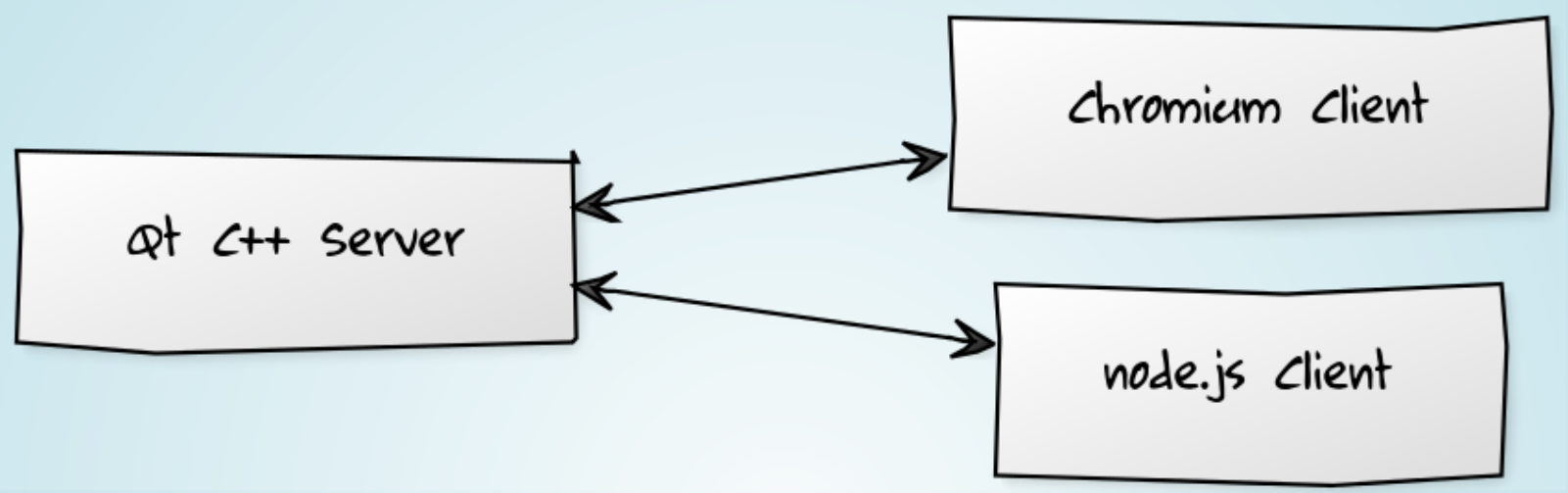
C++ SERVER

```
// setup WebSocket server
QWebSocketServer server(QStringLiteral("QWebChannel Server"),
                        QWebSocketServer::NonSecureMode);
server.listen(QHostAddress::LocalHost, 12345);

// wrap WebSocket clients in QWebChannelAbstractTransport objects
// see: qtwebchannel/examples/webchannel/standalone
WebSocketClientWrapper clientWrapper(&server);

// setup the channel and connect to WebSocket clients
QWebChannel channel;
QObject::connect(&clientWrapper,
                 &WebSocketClientWrapper::clientConnected,
                 &channel, &QWebChannel::connectTo);
```

JAVASCRIPT CLIENT

```
var socket = new WebSocket("ws://localhost:1234");
socket.onopen = function() {
    new QWebChannel(socket, function(channel) { /* ... */ });
}
```

DEMO

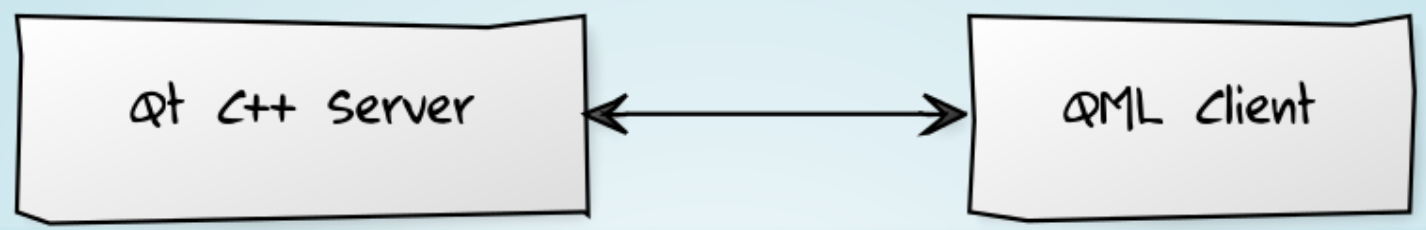


QT QUICK 2 CLIENT

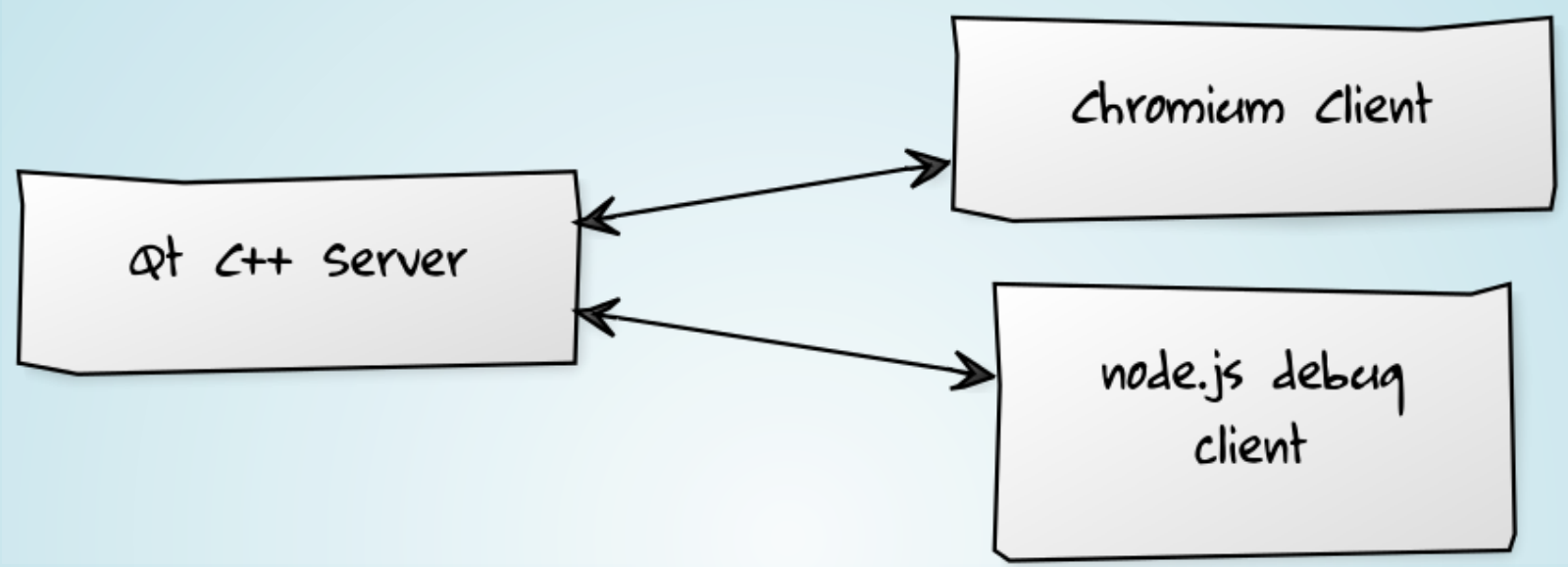
```
import QtWebChannel 1.0
import "qrc:///qtwebchannel/qwebchannel.js" as JSClient
import Qt.WebSockets 1.0
```

```
WebSocket {
    id: socket
    url: "ws://127.0.0.1:12345"
    // mimick HTML WebSocket API
    function send(message) { sendTextMessage(message); }
    property var onmessage;
    onTextMessageReceived: { onmessage({data:message}); }
    // create WebChannel once connected
    onStatusChanged: if (socket.status == WebSocket.Open) {
        new JSClient.QWebChannel(socket, webChannelInitialized);
    }
    // try to connect to server
    active: true
}
```

DEMO



DEMO



TIPS & TRICKS

minimize IPC traffic

- share a single WebChannel with multiple WebViews
- block updates if WebViews are invisible
- only publish "safe" objects
- keep memory overhead in mind

QT WEBCHANNEL

- easy to use
- very versatile
- asynchronous API
- serialization overhead

OUTLOOK

- Qt WebEngine
- Qt WebView for Android, iOS
- further simplify usage
- improved multi-client support
- more language bindings

THE END

QUESTIONS? FEEDBACK?

Milian Wolff / www.kdab.com

```
git clone git@gitorious.org:qt/qtwebchannel.git
```