How Qt Helped RIM Build BlackBerry 10

Jeff Kehres, Research In Motion

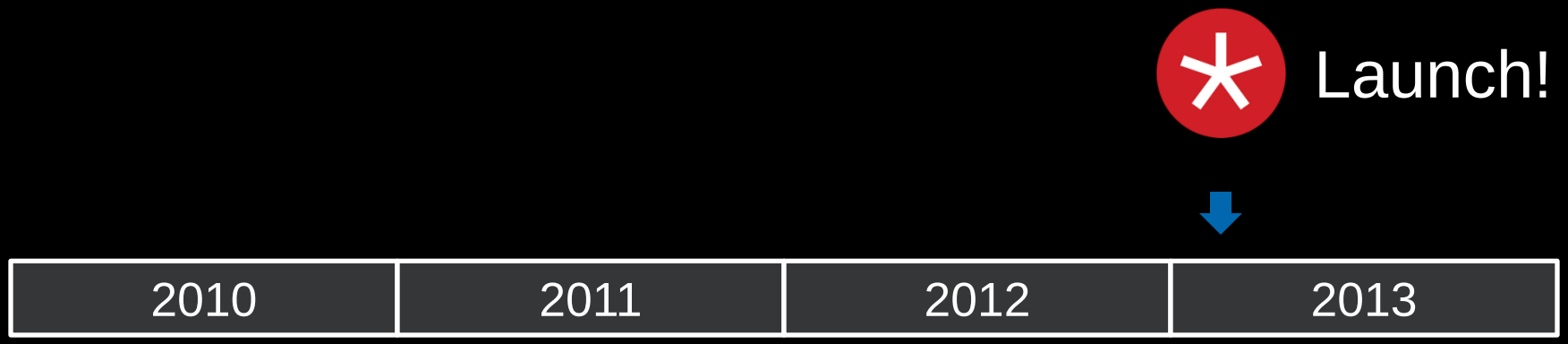
November 19, 2012

Overview

- How did we get here?
- What is BlackBerry 10?
- What role does Qt play?
- What's next?

A Brief History

BlackBerry 10 Timeline



Reinventing BlackBerry

- Needed an OS for the next decade
- Requirements:
  - Rock solid kernel that supports multi-core
  - Open standards and technologies
  - Easy to write/port apps
  - One OS for all devices
  - Amazing user experience

BlackBerry 10 Timeline

QNX

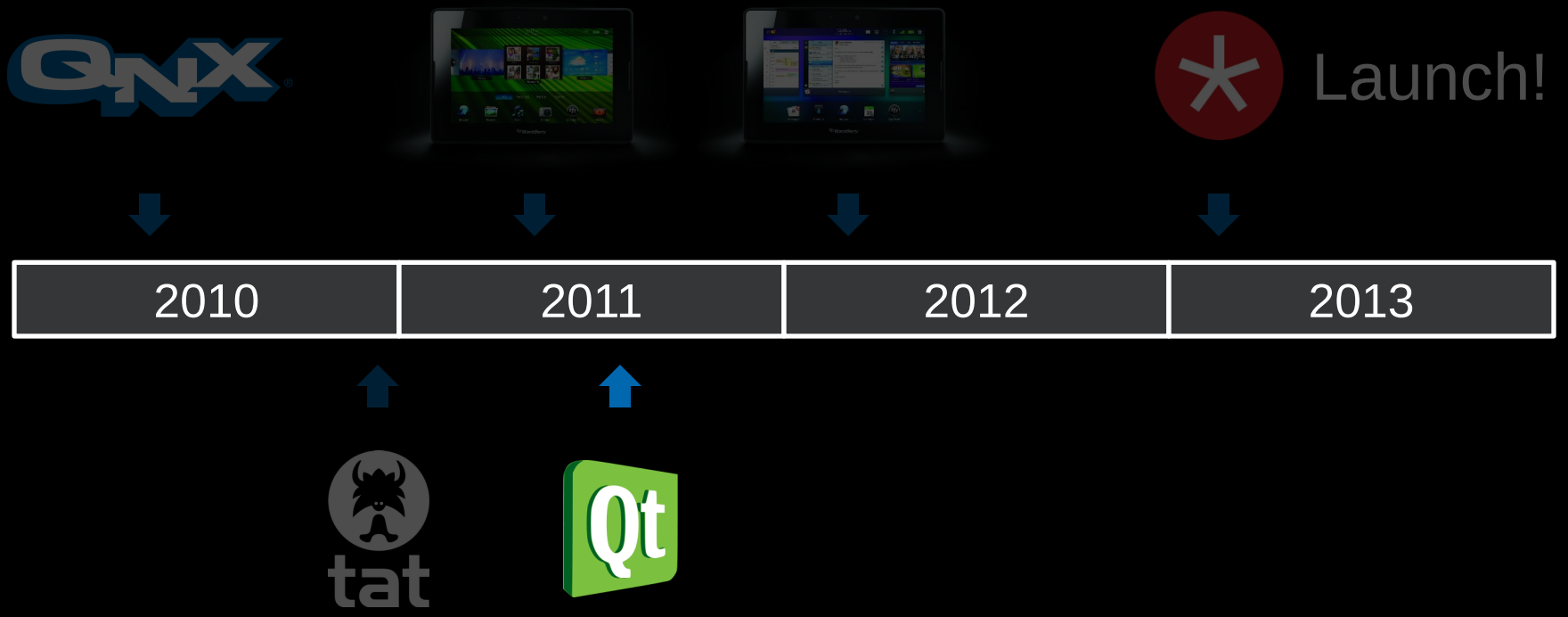
- Real-time operating system (RTOS)
- Scalable: embedded → parallel computing
- Certifications:
  - POSIX
  - Security (EAL4+)
  - Safety (IEC 61508 – SIL3)
- BlackBerry Tablet OS

BlackBerry 10 Timeline

TAT: The Astonishing Tribe

- Amazing team of designers and developers
- Passionate about UI/UX
- Worked with many major players in mobile
- Technology:
  - Declarative UI framework
  - Rendering engine

BlackBerry 10 Timeline



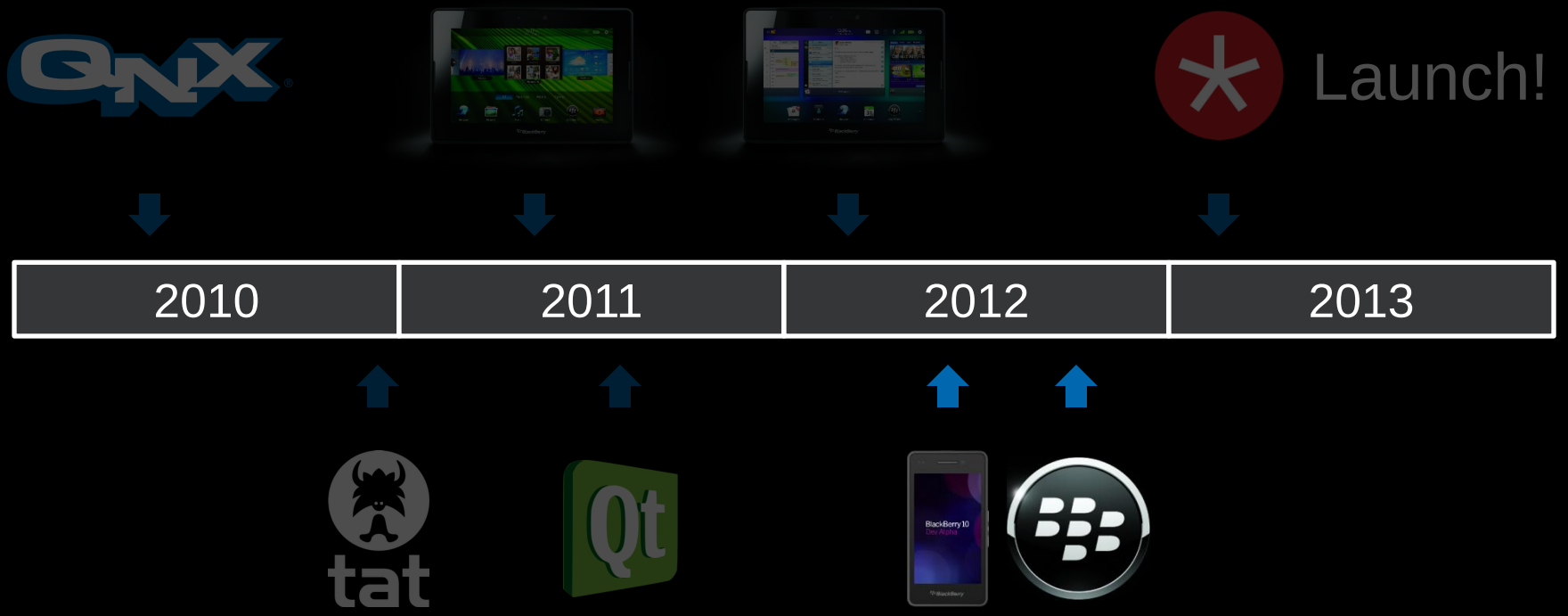
Qt, QNX, and Me

- Started as a side project
- Previous port of Qt/Embedded for QNX
- New port of Qt 4.7 to PlayBook using QPA
- Key tasks:
  - Cross compiling
  - QPA plugin
- Use Unix backend beyond QPA

Qt on BlackBerry Today

- Maintained by dedicated team and KDAB
- Platforms: BlackBerry 10 & Tablet OS 1.x/2.x
- Versions: 4.8, 5.0
  - Qt 4.8.4 shipping on BlackBerry 10!
- Source code: http://qt.gitorious.org/qt
- Wiki: http://qt-project.org/wiki/BlackBerry

BlackBerry 10 Timeline



A New Mobile Computing Platform

BlackBerry 10 Platform

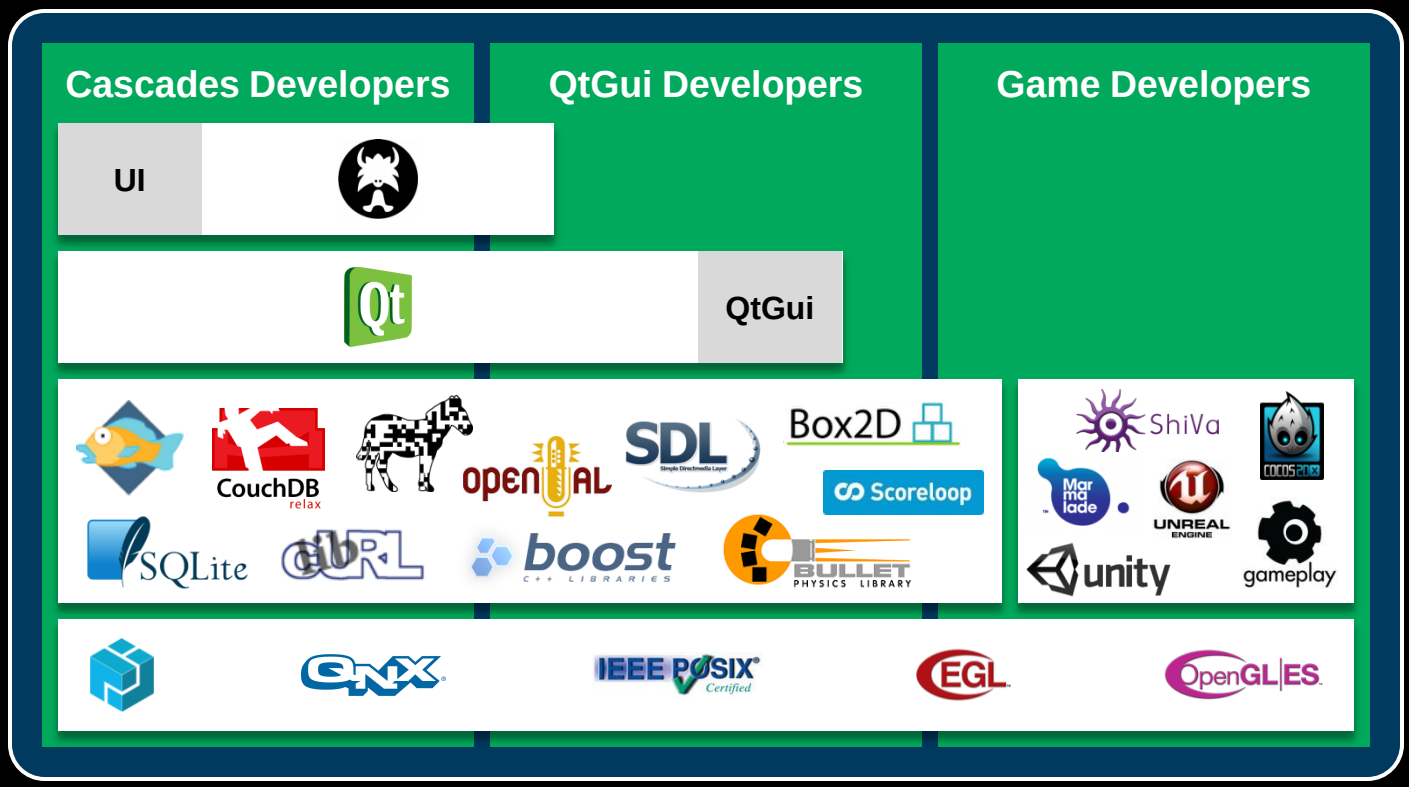
(HTML5)
WebWorks™

(C/C++/Qt)
Native

(ActionScript)
Adobe® AIR®

(Java)
Android™
Runtime

Native Platform



Cascades UI Framework

- Signature look and feel of BlackBerry
- Backend:
  - Optimized for BlackBerry hardware
  - Separate thread for rendering & animations
- C++ and QML interface
- Mutually exclusive with QtGui / Qt Quick


```
#include <QApplication>
#include <QDeclarativeView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QDeclarativeView view;
    view.setSource(QUrl::fromLocalFile("app/native/assets/qtquick.qml"));
    view.showFullScreen();
    return app.exec();
}
```

```
////////////////////////////////////////////////////////////////////////////////
```

```
#include <bb/cascades/Application>
#include <bb/cascades/QmlDocument>
#include <bb/cascades/Page>

using namespace bb::cascades;
int main(int argc, char **argv)
{
    Application app(argc, argv);
    QmlDocument *qml = QmlDocument::create("asset:///cascades.qml");
    Page *page = qml->createRootObject<Page>();
    app.setScene(page);
    return app.exec();
}
```

```cpp
#include <QApplication>
#include <QDeclarativeView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QDeclarativeView view;
    view.setSource(QUrl::fromLocalFile("app/native/assets/qtquick.qml"));
    view.showFullScreen();
    return app.exec();
}
```


```cpp
////////////////////////////////////////////////////////////////////////////////

#include <bb/cascades/Application>
#include <bb/cascades/QmlDocument>
#include <bb/cascades/Page>

using namespace bb::cascades;
int main(int argc, char **argv)
{
    Application app(argc, argv);
    QmlDocument *qml = QmlDocument::create("asset:///cascades.qml");
    Page *page = qml->createRootObject<Page>();
    app.setScene(page);
    return app.exec();
}
```

```cpp
#include <QApplication>
#include <QDeclarativeView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QDeclarativeView view;
    view.setSource(QUrl::fromLocalFile("app/native/assets/qtquick.qml"));
    view.showFullScreen();
    return app.exec();
}
```

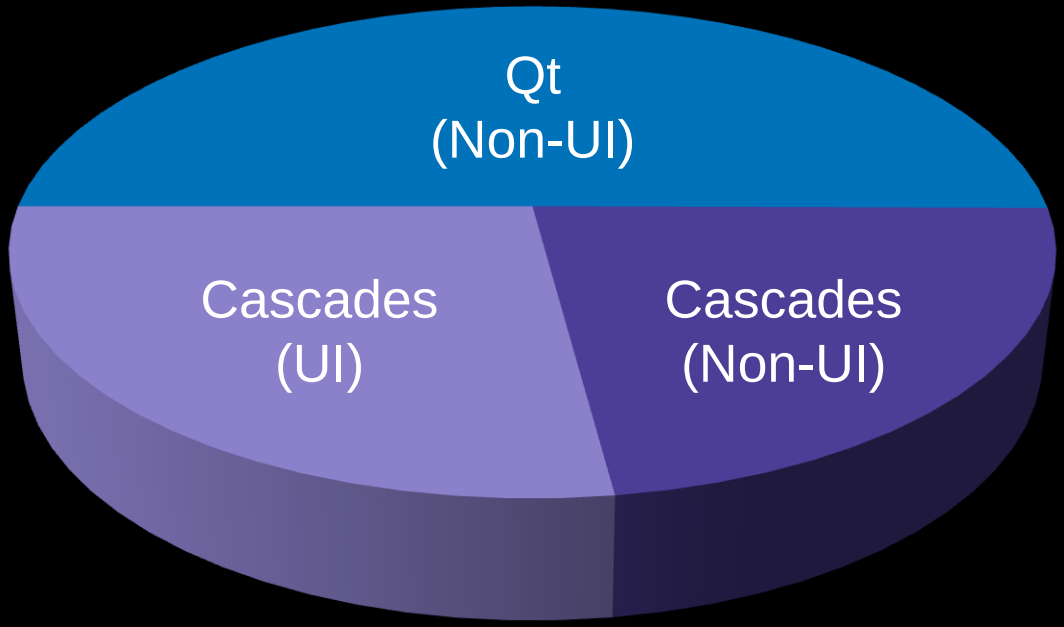

```cpp
////////////////////////////////////////////////////////////////////////////////

#include <bb/cascades/Application>
#include <bb/cascades/QmlDocument>
#include <bb/cascades/Page>

using namespace bb::cascades;
int main(int argc, char **argv)
{
    Application app(argc, argv);
    QmlDocument *qml = QmlDocument::create("asset:///cascades.qml");
    Page *page = qml->createRootObject<Page>();
    app.setScene(page);
    return app.exec();
}
```

```
import QtQuick 1.0

Text {
    text: "Hello world!"
}
```


```
////////////////////////////////////////////////////////////////////////////////

import bb.cascades 1.0

Page {
    content: Label {
        text: "Hello world!"
    }
}
```

```
////////////////////////////////////////////////////////////////////////////////

Page *page = new Page;
Label *label = new Label;
label->setText("Hello world!");
page->setContent(label);
Application::instance()->setScene(page);
```

API Breakdown for Cascades Developers



Qt 4.8 Modules

- QtCore
- QtDeclarative*
- QtGui*
- QtMultimedia*
- QtNetwork
- QtOpenGL*
- QtScript
- QtScriptTools*
- QtSql
- QtSvg*
- QtTest
- QtXml
- QtXmlPatterns

* Contains APIs not compatible with Cascades UI

Other Qt Modules

- QtMobility 1.2
  - QtMultimediaKit*
  - QtSensors
  - QtLocation → QtLocationSubset
  - QtConnectivity → QtNfcSubset
- Qt Labs
  - QCollator → QtCollator

* Contains APIs not compatible with Cascades UI

QtLocationSubset

- Positioning and geocoding APIs only
- Backported “text” property of QGeoAddress from Qt 5
- Added extra settings via properties of sub-classes
- QtMobilitySubset namespace
- Source code: http://blackberry.github.com

QtNfcSubset

- NDEF encoding/decoding API only
- Added support for Smart Poster records
- Fixed bug in binary message format
- QtMobilitySubset namespace
- Source code: http://blackberry.github.com

Cascades Modules (Non-UI APIs)

- bb
- bb::data
- bb::device
- bb::multimedia
- bb::network
- bb::pim
- bb::platform
- bb::platform::bbm
- bb::system
- bb::utility
- bb::utility::i18n

Cascades Modules (UI APIs)

- bb::cascades
- bb::cascades::advertisement
- bb::cascades::maps
- bb::cascades::multimedia
- bb::cascades::pickers
- bb::cascades::places

Cascades Conventions

- Follow Qt conventions, except:
  - Namespaces, no “Q” prefix
  - Virtual keyword for overridden methods
  - Enums in standalone classes
  - Builder pattern (primarily for UI, optional)
- Expose APIs in QML, when possible
- Namespace → library name, include path
  - bb::data → libbbdata, <bb/data/Foo>

```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```cpp
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

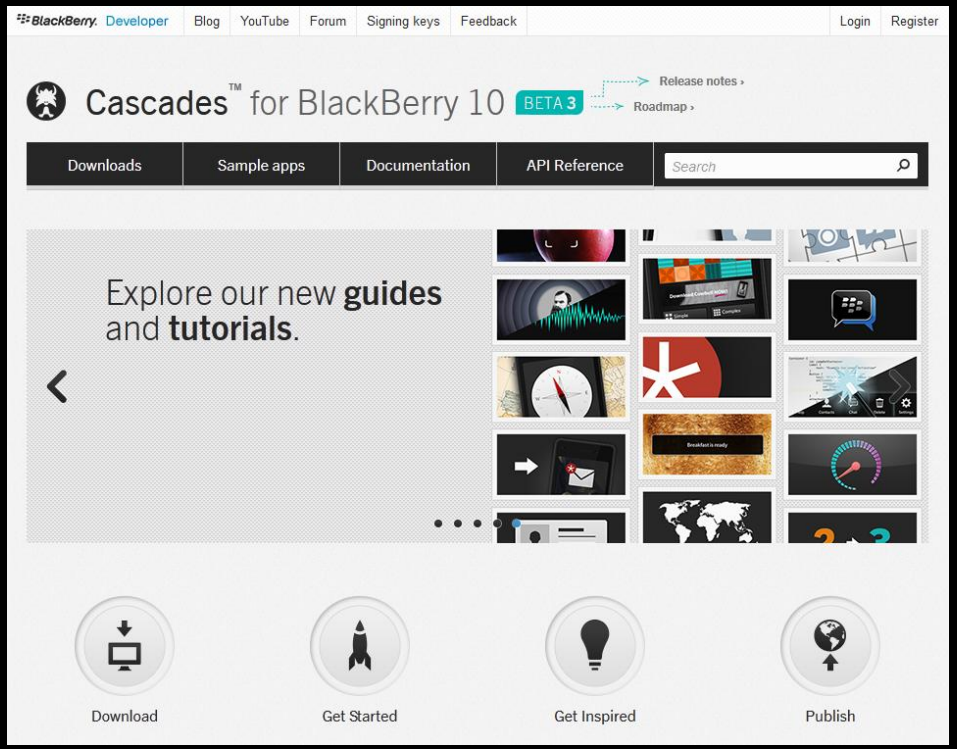
```
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

```cpp
#include <bb/device/SdCardState>

namespace bb {
namespace device {

class BB_DEVICE_EXPORT SdCardInfo : public QObject {
    Q_OBJECT
    Q_PROPERTY(bb::device::SdCardState::Type state READ state
               NOTIFY stateChanged FINAL)
    Q_PROPERTY(bb::device::SdCardActivity::Type activity READ activity
               NOTIFY activityChanged FINAL)
public:
    explicit SdCardInfo(QObject *parent = 0);
    virtual ~SdCardInfo();
    bb::device::SdCardState::Type state() const;
    bb::device::SdCardActivity::Type activity() const;
Q_SIGNALS:
    void stateChanged(bb::device::SdCardState::Type newState);
    void activityChanged(bb::device::SdCardActivity::Type newActivity);
private:
    SdCardInfoPrivate * const d_ptr;
    Q_DECLARE_PRIVATE(SdCardInfo)
    Q_DISABLE_COPY(SdCardInfo)
};

} // namespace device
} // namespace bb
```

For More Details…

https://developer.blackberry.com/cascades/

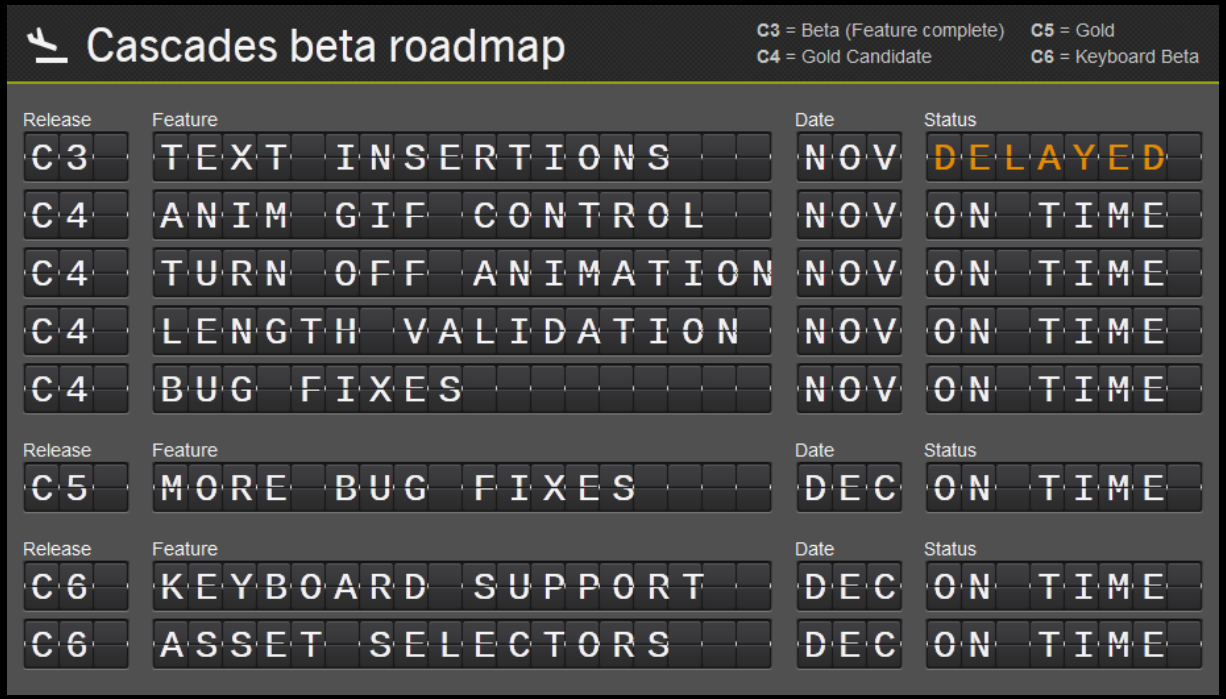
- Documentation
- Samples
- SDK
- Simulator



A Look Ahead

Cascades Roadmap

https://developer.blackberry.com/cascades/download/roadmap/




BlackBerry®
Qt

Some of Our Ideas for Qt

- Don’t assume QtGui for UI
  - Minimize dependencies on QtGui
  - Move some low-level classes out of QtGui
- More flexibility for creating QML bindings
- Fix limitations of meta-object compiler (moc)
- Unify strategy for platform abstraction
- Keep libraries small and focused

Next Steps

- Launch BlackBerry 10!
- Start thinking about Qt 5
- Engage more with Qt community
- Contribute changes beyond BlackBerry port
- Help steer evolution of Qt

Questions?