Qt World Summit 2016
October 18-20 | San Francisco, USA

Qt for iOS A to Z

Mike Krus, Senior Software Engineer at KDAB

Qt on iOS

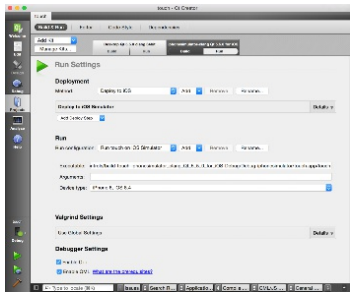
- Recent version of Qt & Creator
  - Qt 5.4
  - Static build
- non-GUI layers of Qt compile as normal
  - Unix / Mach backend for files, sockets, memory
- QPA layers maps `QWindow` to `UIView`
- Widget-based UI possible
  - Not recommended
- QtQuick UI rendered using OpenGL ES (2 or 3)
- iOS 8 and up (for Qt 5.5)

Requirements

- A Mac
  - Basic laptop or Mac Mini sufficient
  - Mac Mini works as headless build machine
  - Not a VM on Not a Mac
- An Apple Id
- An Apple developer subscription
  - Applies to more than one Apple ID
  - No limit on team size
  - iOS 9 introduce 'Personal Teams'
- Xcode

Demo

Time to try Creator...



QMake

`qmake` generates

- Makefiles
- `MyProject.xcodeproj`
- `Info.plist`
- qml files, contained in a qrc
- `main.cpp`
- `myproject_plugin_import.cpp`
- `myproject_qml_plugin_import.cpp`

Anatomy of an App

- Application is a *bundle*
  - Specially structured directory
    - *Documents, Library/Cache...*
  - Not a compressed zip/jar
- Bundles contain an `Info.plist` XML file
  - Metadata about author, supported platforms, architectures, copyright
  - Specifies executable to launch inside the bundle
  - File-types / URLs / mime-types
- Bundle has a unique identifier (reverse DNS style), version number, build number
- Also contains arbitrary resources as plain files

QMake Variables

- `darwin/mac` vs `osx` vs `ios`
- Variables

```
QMAKE_IOS_DEPLOYMENT_TARGET = 8.0
QMAKE_IOS_TARGETED_DEVICE_FAMILY = 2
QMAKE_IOS_DEVICE_ARCHS = armv7 arm64
QMAKE_IOS_SIMULATOR_ARCHS = i386 x86_64

VERSION = 1.2.3
BUILDID = 55

plist.input = Info.plist.in
plist.output = $$OUT_PWD/Info.plist
QMAKE_SUBSTITUTES += plist
QMAKE_INFO_PLIST = $$OUT_PWD/Info.plist
```

- Qt 5.8: `uikit, macos, tvos, watchos, QMAKE_APPLE_...`

PList Template

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
        "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>CFBundleIdentifier</key>
    <string>com.kdab.${PRODUCT_NAME:rfc1034identifier}</string>
    <key>CFBundleDisplayName</key>
    <string>${PRODUCT_NAME}</string>
    <key>CFBundleName</key>
    <string>${PRODUCT_NAME}</string>
    <key>CFBundleShortVersionString</key>
    <string>$$VERSION</string>
    <key>CFBundleVersion</key>
    <string>$$VERSION.$$BUILDID</string>
    <key>LSRequiresIPhoneOS</key>
    <true/>
    <key>UISupportedInterfaceOrientations</key>
    <array>
        <string>UIInterfaceOrientationLandscapeLeft</string>
        <string>UIInterfaceOrientationLandscapeRight</string>
    </array>
    <key>UIFileSharingEnabled</key>
    <true/>
    ...
</dict>
</plist>
```


Icons & Splash Screens - Resources

- Include *properly named files* in your bundle

```
meta.files = $$files($$PWD/meta/*)
QMAKE_BUNDLE_DATA += meta
```

- Reference in `Info.plist`

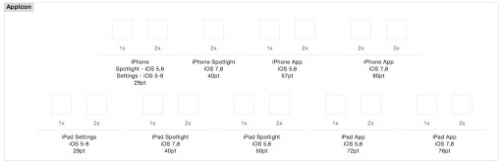
```
...
<key>CFBundleIcons</key>
<dict>
    <key>CFBundlePrimaryIcon</key>
    <dict>
        <key>CFBundleIconFiles</key>
        <array>
            <string>Icon-57.png</string>
            <string>Icon-72.png</string>
            <string>Icon-72@2x.png</string>
        </array>
        <key>UIPrerenderedIcon</key>
        <true/>
    </dict>
</dict>
...
```

Icons & Splash Screens - Asset Catalogs

- Include *asset bundles* in your bundle

```
meta.files = $$files($$PWD/meta/*.xcassets)
QMAKE_BUNDLE_DATA += meta
```

- Edit in Xcode



Remember you're working on a copy!

Getting to know Xcode

- Don't be afraid of Xcode
- Entrypoint to dedicated editors
- Manages developer account
- Manages devices and applications
- Builds & deploys your code
- Archive management
- SDK documentation and examples
- Excellent debugger

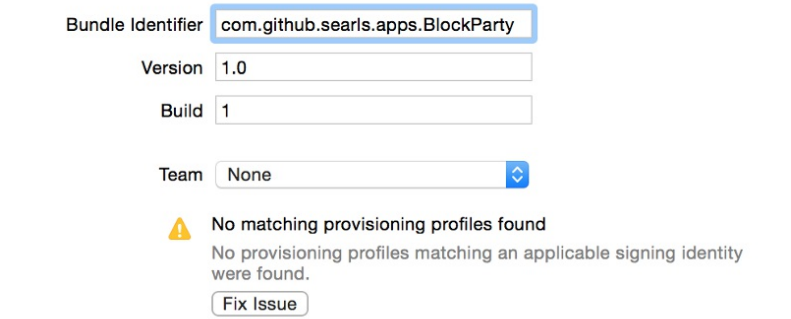
The Simulator

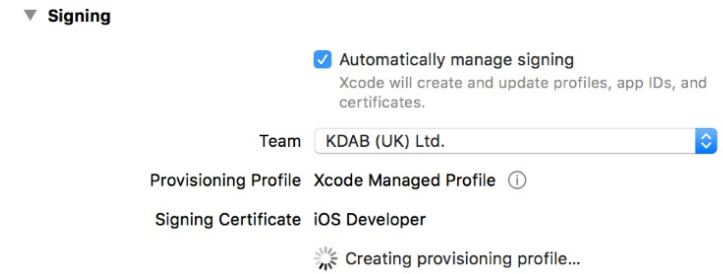
The simulator is *not* an emulator

- Application compiled for x86 or x64
- Links against custom frameworks providing iOS APIs
- Simulator frontend allows hardware emulation
- Runtime environment
- Differences from real hardware
  - Code-signing, OpenGL performance, sensors

Running on real hardware

- **You must run your app on real hardware**
- iOS applications must *always* be **code signed**
  - Certifies origin of the app
  - Prevents tampering
- **Entitlements**
  - Enables some features, iCloud, Health, Push Notifications...
- Deployment requires **Provisioning Profiles**
  - Controls where app can run
  - Developer profiles
  - AppStore or Ad-Hoc profiles



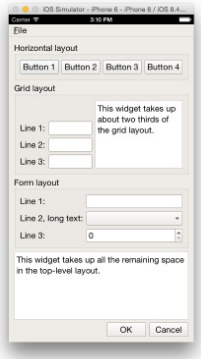


Sandboxing

- iOS never exposes the filesystem to the user
- At runtime, applications can only access files within their 'sandbox'
  - Implemented as kernel access control (ACL), not a chroot
- No access to other application's data
- Other restrictions
  - No use of private APIs
    - Checked mechanically by tooling
  - No downloading code or JIT-ing
    - Attempts to mark data as executable will fail
- Application groups in iOS 8
  - Apps with matching group IDs can share containers
  - Group IDs defined in the member center, registered in the provisioning

Qt UI - Widgets

Widgets work, but...



... don't

QML - 1

- QML
  - UI rendered using OpenGL
  - Designer friendly
  - Animations
- Quick.Controls, Quick.Dialogs
  - More layouts
  - `ComboBox`
  - `StackView`
  - `FileDialog`
    (`fileDialog.folder: fileDialog.shortcuts.pictures`)
  - Text Input
    - Selection is different *(much better in 5.7!)*
    - Keyboard avoidance is a pain (see `Qt.inputMethod.hide()`)

QML - 2

- Retina & resources
  - Coordinates are in `qreal`, *points NOT pixels*
  - Set `border.pixelAligned: false`
- Image sources support `@2x`

```
Image { src: "/images/foo.png"; width: 200; height 200 }
```

- If present, will load `foo@2x.png` (400x400 px) on retina devices
- Embrace this, don't fight it by playing with size and scale of root item!

QML - 3

- Plain QML has no visual style
- Controls1 look like widgets
- Controls2 no iOS look
- Native appearance is problematic
  - Define native!
  - Moving target: Apple restyle UIKit across OS upgrades

Qt Demo /Users/Shared/Qt/Examples/Qt-5.7/quickcontrols/controls/touch

Qt Demo /Users/Shared/Qt/Examples/Qt-5.7/quickcontrols2/gallery

UIKit idioms

- UIKit provides standard high-level view structures
  - Paged, tabbed, master / detail
  - User and designers familiar with the behaviour and visual presentation
- Also supports basic navigation and transitions
  - Push / Pop navigation
  - Modal presentation
- Absence of pop-up / pop-over UI
  - Modal yes / no dialogs the (infrequent) exception
- Transitory modal UI for rotary wheels
- Powerful animation engine

Handling Views

- Hard to not load everything at start up
- Loader useful but destructive, can't transition
- Using components

```
var component = Qt.createComponent("foo.qml")
if (component.status == Component.Ready) {
    newView = component.createObject(parent, {opacity: 0, "anchors.fill": parent})

    previousView = currentView
    currentView = newView
    if (previousView) {
        previousView.viewWillDisappear()
        fadeOut.target = previousView
        fadeOut.running = true
    }
    currentView.viewWillAppear()
    fadeIn.target = currentView
    fadeIn.running = true
}
```


Other UI issues

- Phone vs tablets
- Portrait vs landscape
- Cross-platform considerations
- Accessibility

Qt Modules

- Sensors
- Camera
- Position (GPS), Location (Maps)
- In-App purchase
- Qt3D
- ...

Qt not enough?

- Coverage not complete
  - File-type association
  - Background operations
  - Sharing
  - iCloud
  - Access to contacts, accounts (Twitter / Facebook)
  - CoreMotion (steps...), MapKit, HealthKit, GameKit, PassKit, Security, PushKit...
  - WebKit!
  - ...
- Through specialised Objective-C APIs
- Not wrapped by Qt, easy to invoke yourself

Software Platform

- Low-level system shared with OS X
  - Mach microkernel
  - BSD userspace libraries (libC, crypto)
- Apple system libraries
  - CoreAudio, CoreAnimation, CoreLocation, AVFoundation, CoreWLAN
- Cocoa Touch
  - Objective-C, like Cocoa
  - Natively designed for mobile & touch
  - UIKit, Standard widgets
  - UI class prefix
  - Many others (MapKit, HealthKit, ...)

Objective-C++

- Objective-C & C++ combined
- Language syntaxes are orthogonal
- `.mm` file extension
- `OBJECTIVE_SOURCES` in *qmake*

Bridging the Objective-C and Qt object systems is possible. Potentially relevant if incorporating a 3rd-party framework for iOS.

Mixing Qt & UIKit - Using QQuickItem

- Derive from `QQuickItem`
- Handle `windowChanged(QQuickWindow*)` signal to create `UIView`
- Handle `visibleChanged()` signal to show/hide
- Overload `geometryChanged(const QRectF &newGeometry, const QRectF &oldGeometry)` to resize

Demo ios/UITextView

Mixing Qt & UIKit - Using QPA

- QPA layer exposes native resources
- Parent UIView hierarchy to main Qt window
- Unsuitable for embedding individual widgets

```
QT += gui-private
```

```
1 QWindow* window = ... ;
2 UIView *view = static_cast<UIView*>(QGuiApplication::platformNativeInterface()->
3     nativeResourceForWindow("uiview", window));
4 Q_ASSERT(view);
5
6 UIViewController* controller = [[view window] rootViewController];
7 Q_ASSERT(controller);
```

- Create a view / controller heirarchy programmatically
- Via a system or 3rdparty library

Application Delegate

Use Objective C categories

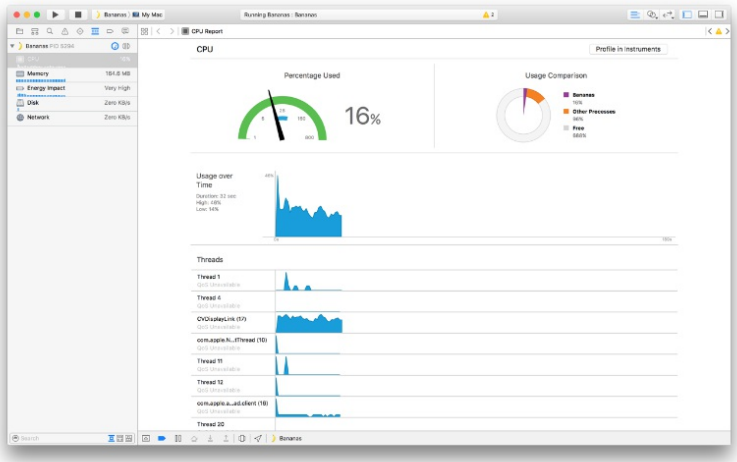
```
@interface QIOSApplicationDelegate
@end

@interface QIOSApplicationDelegate(MyApp)
@end

@implementation QIOSApplicationDelegate(MyApp)
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application
{
    qDebug() << "Lets release some memory before we get killed";
}

- (BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
{
    ...
}
```


Application Delegate

Use own delegate

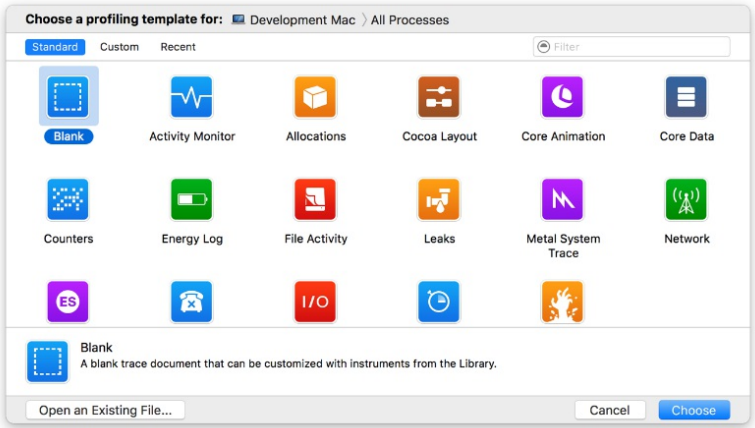
```
@interface MyAppDelegate : UIResponder <UIApplicationDelegate, DBSessionDelegate>
+(MyAppDelegate *)sharedAppDelegate;
@end

@implementation MyAppDelegate
static MyAppDelegate *sharedAppDelegate = nil;
+(MyAppDelegate *)sharedAppDelegate {
    static MyAppDelegate *shared = nil;
    if (!shared)
        shared = [[MyAppDelegate alloc] init];
    return shared;
}

- (BOOL)application:(UIApplication *)application
        willFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    return YES;
}
@end

int main(int argc, char *argv[]) {
    ...
    [[UIApplication sharedApplication] setDelegate:[MyAppDelegate sharedAppDelegate]];
    ...
}
```


Debugging & Profiling



(c) Apple Inc.

Debugging & Profiling



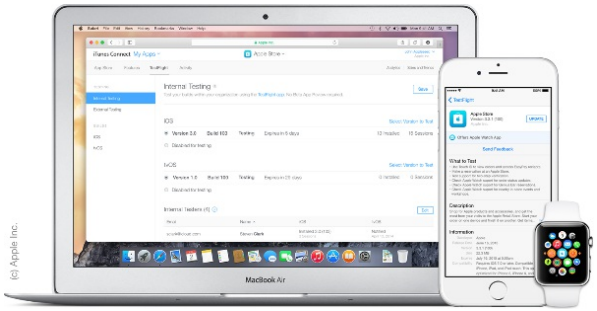
(c) Apple Inc.

Common Issues

- QML / JS: slow startup, no JIT compiler
- Memory allocation
  - The `applicationDidReceiveMemoryWarning:` method of your app delegate.
  - The `didReceiveMemoryWarning` method of your UIViewController classes.
  - The `UIApplicationDidReceiveMemoryWarningNotification` notification.
- Graphical effects...

External Testing

Test Flight



- Hockey App, AWS Device Farm...

Thank you!

mike.krus@kdab.com - www.kdab.com

KDAB