



# ArcGIS Runtime SDKs: Building Cross-Platform Apps

Tyler Schiewe  
Lucas Danzinger  
Rich Zwaap  
Rex Hansen

SEE  
WHAT  
OTHERS  
CAN'T

# Agenda

- Cross-platform review
- ArcGIS Runtime cross-platform options
  - Java
  - Qt
  - .NET



# Native vs Web?

- **Native apps**

- App installed on the device
- Use Platform / Operating System APIs
- Best performance and device integration
- Support for connected and offline workflows
- Work well when you have the ability to determine devices
- Use [ArcGIS Runtime SDKs](#) to create native apps

- **Web apps**

- Web site/app downloaded from a server
- Best for wider range of users on unknown devices
- Use the [ArcGIS API for JavaScript](#) to create web client solutions

- User experience and capabilities increasingly blurred as technologies evolve

<http://esriurl.com/ChoosingAnAPI>

# Native app cross-platform considerations

- Benefits

- Share application code
- Enforces good design patterns
- Makes your app available to more users

- Challenges

- Testing
- User experience of your app may vary
- Handling platform idiosyncrasies (security, bugs, etc)
- Development cost



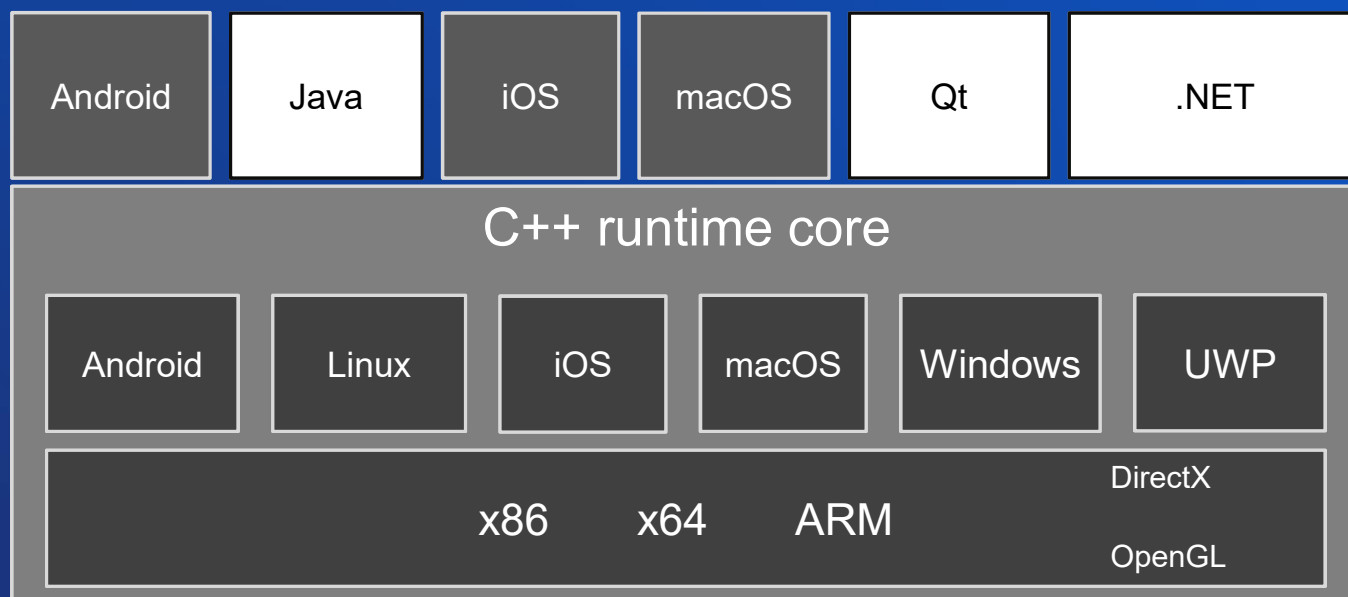
# Building Native Apps on Multiple Platforms

- How do you choose a cross platform SDK?
  - Business, technical, and user requirements
  - Developer skillset
- Multiple options available
  - Java
  - Qt
  - .NET/Xamarin



# ArcGIS Runtime cross-platform options

- All Runtime APIs built on common Runtime core



# Java

Tyler Schiewe

# Qt

Lucas Danzinger

# .NET and Xamarin

Rich Zwaap



# Java

Tyler Schiewe





## Cross platform Java Development

- “Write once, run anywhere”
- OpenJDK is free
- JavaFX for building modern desktop apps with a native look and feel
- Lots of free IDEs to choose from
- Massive ecosystem of mature, open-source libraries to use

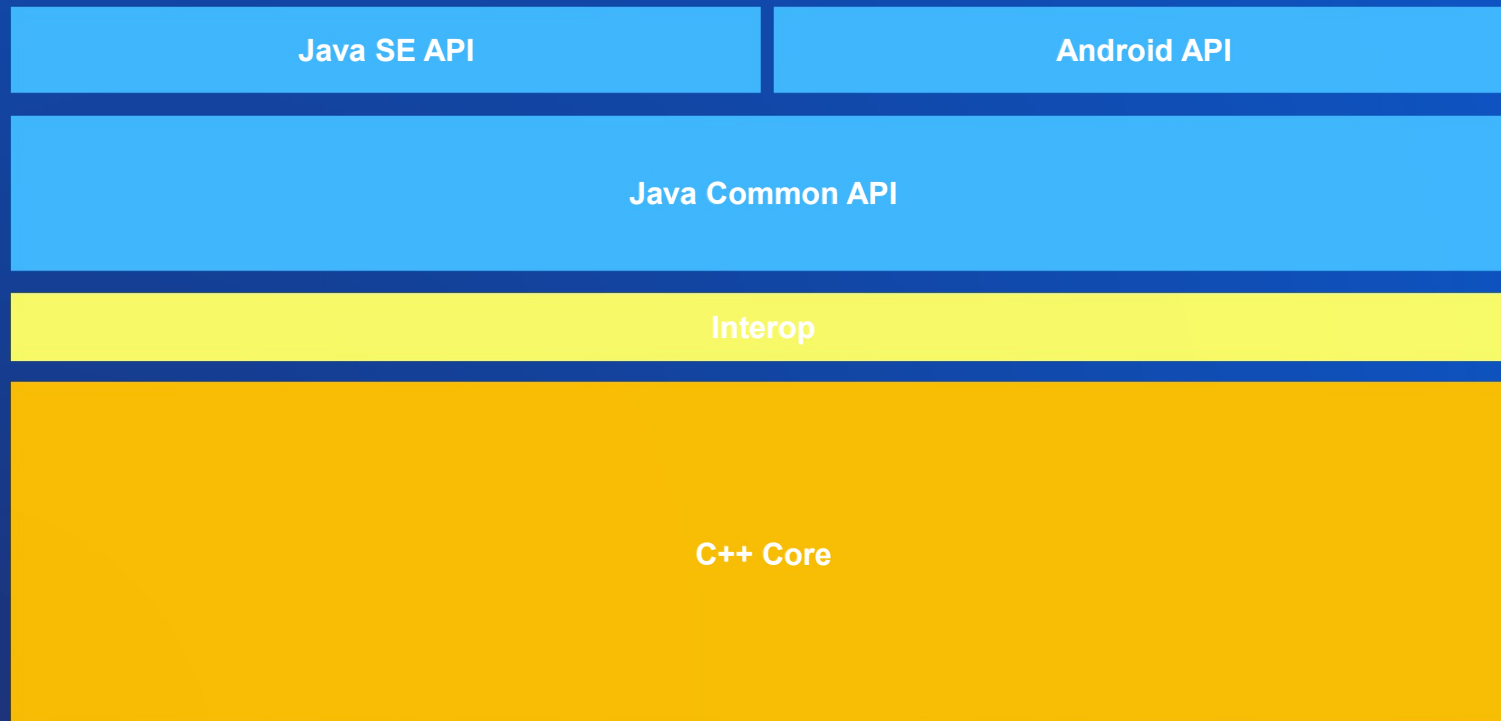


## ArcGIS Runtime SDK for Java

- ArcGIS Runtime SDK for Java targets Windows, Mac, and Linux desktops
- Sits on the ArcGIS Runtime core architecture (C++) via JNI
- Provides MapView and SceneView JavaFX controls



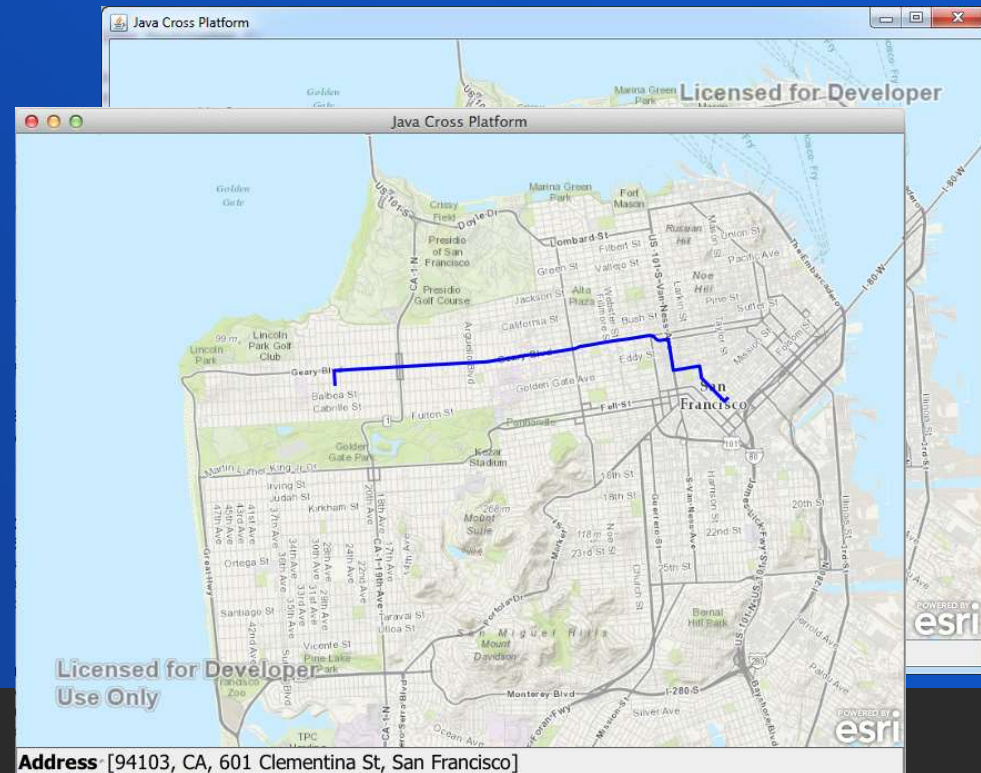
# API Architecture



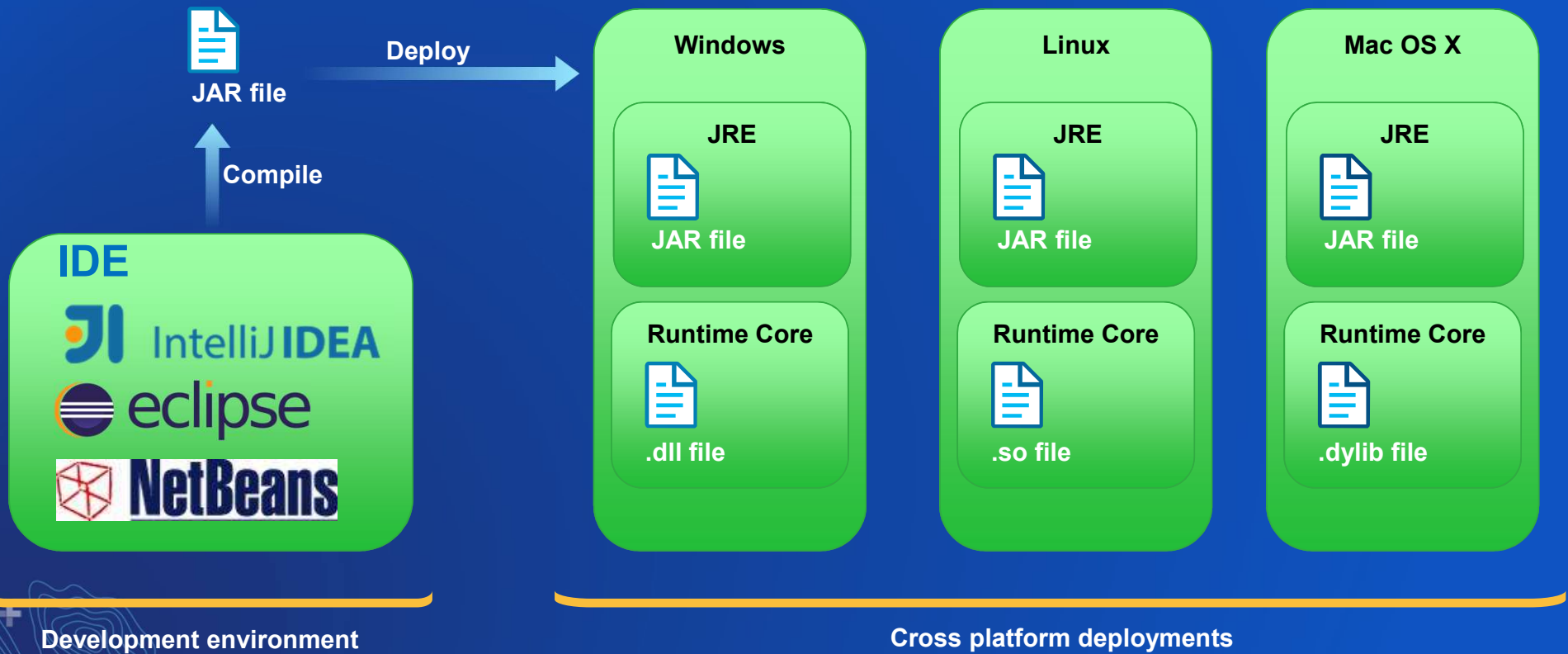
# JavaFX

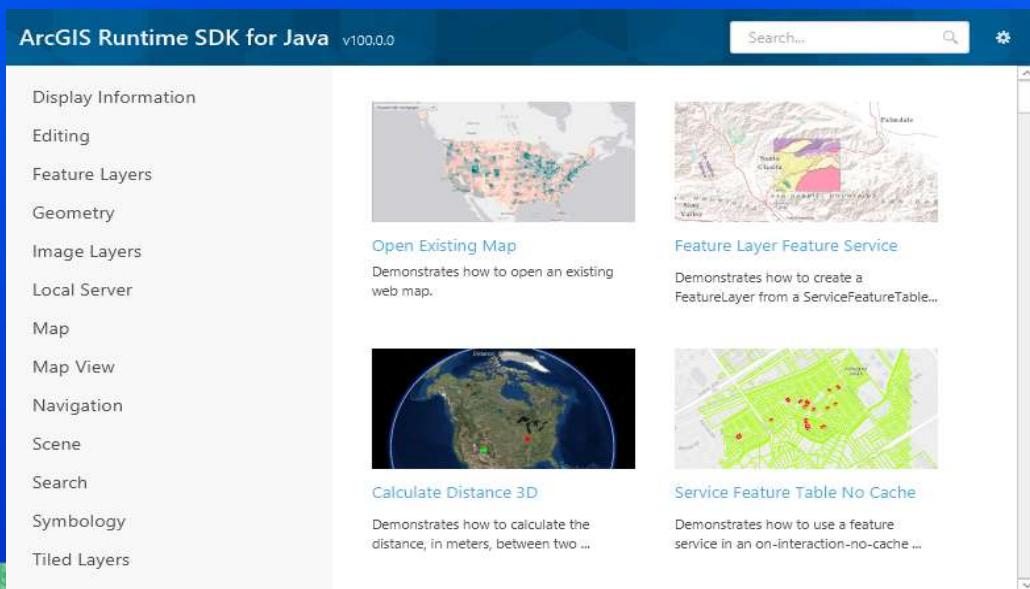
- App launched in a native window
- Styling and theming with CSS
- Programmatic and markup (FXML) options for creating layout

```
<StackPane fx:controller="com.esri.samples.mysample.SampleController"
  xmlns:fx="http://javafx.com/fxml" stylesheets="/css/style.css">
  <MapView fx:id="mapView"/> <!-- SDK control -->
  <HBox StackPane.alignment="TOP_CENTER" maxWidth="200" maxHeight="50" spacing="5" styleClass="panel-region">
    <Label text="Click me: "/>
    <Button fx:id="myButton" onAction="#myEvent"/>
  </HBox>
</StackPane>
```



# Distribution





# Demo

An app for Windows, Linux, and Mac

# Summary

- **Pros**

- Tools are free for commercial use
- Deployments can be identical for ALL platforms
- JavaFX apps style for the platform



- **Cons**

- Clients must have Java installed\*
- Not targeted for mobile or web apps



Qt

Lucas Danzinger





## Agenda (Qt)

- What is Qt?
- Which platforms can I build for?
- How do I get set up?
- What language do I use?
- What will my apps look like?
- What are the Pros and Cons?



# What is Qt?

The Qt Company – [www.qt.io](http://www.qt.io)

*Code less, create more, deploy everywhere*

- Write once, deploy everywhere
- A complete cross-platform software framework
  - C++ libraries
  - Ready-made UI elements
  - Tooling
- Over 1 million developers worldwide
- Open-source community

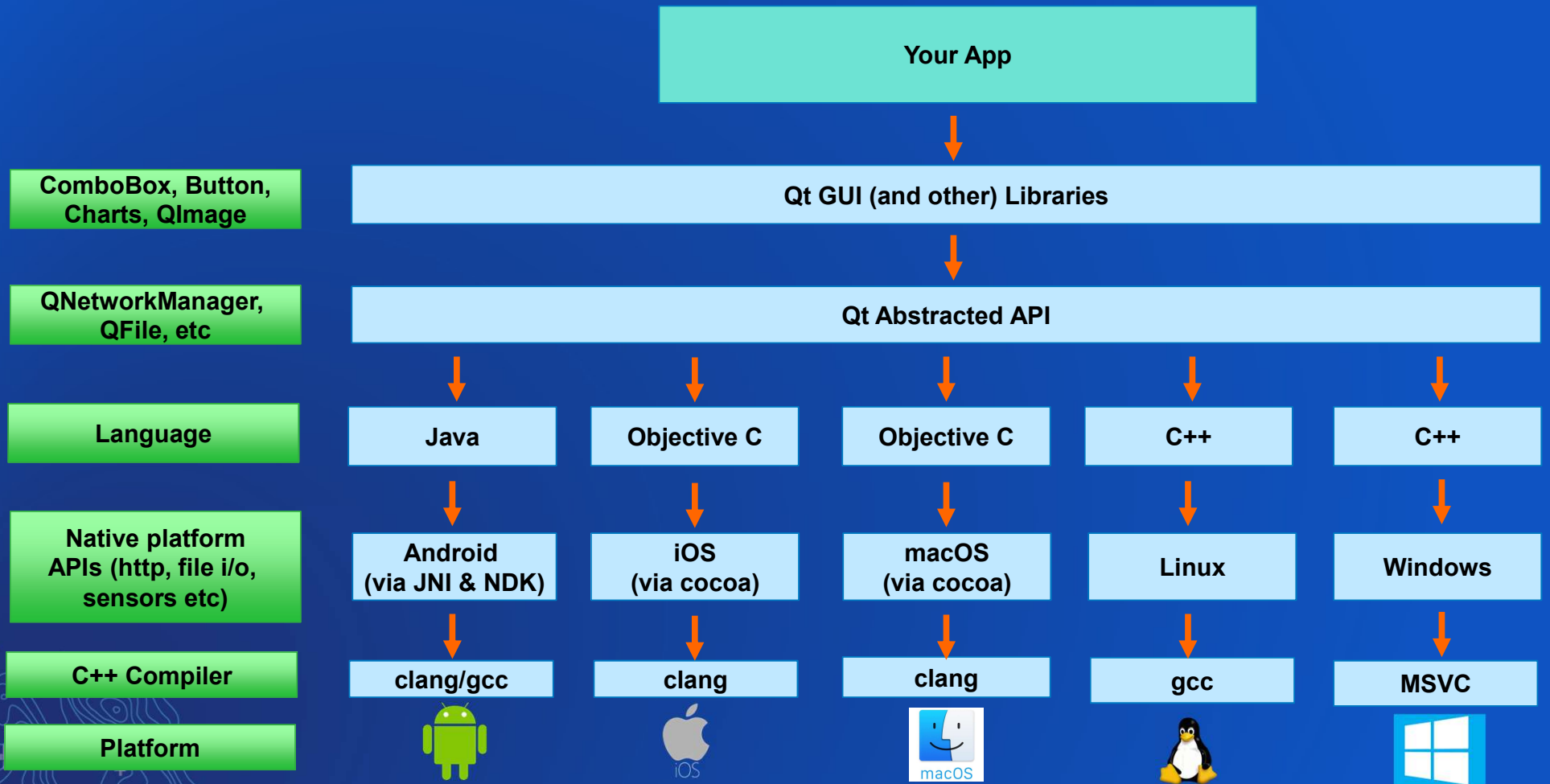


## Which platforms can I build for?

- Windows – x86, x64
- Linux – x64, arm 64 (beta)
- macOS – x64
- Android – armv7, armv8, x86
- iOS – arm64, sim



# How does it work?



# Setup

- Setup Qt:
  - Create account
  - License / open-source?
  - Install kits
- Setup ArcGIS Runtime SDK for Qt
  - <https://developers.arcgis.com/qt/latest/>
- Compiler, SDK dependencies
  - macOS/iOS: Xcode compiler
  - Windows: Visual Studio compiler, debugging tools
  - Linux: GCC compiler
  - Android: Android NDK and SDK
- IDE

Qt Creator



# What language will I use?

## 2 APIs: same Runtime Core (C++)

- C++ API
  - Modern C++ language (C++ 11)
  - Fast performance
  - DSA open source app built with this
- QML API
  - Easy to use & learn
  - Imperative JavaScript business logic code
  - AppStudio (Survey 123)



# QML API

Example QML API code

Highly readable JSON/CSS-like syntax

Declarative UI elements

Imperative JavaScript Code to handle events

```
Rectangle {
  MapView {
    id: mv
    anchors.fill: parent
    Map {
      id: map
      BasemapStreetsVector {}
    }
  }
  Button {
    anchors {
      left: parent.left
      top: parent.top
    }
    text: "Zoom to Hawaii"
    enabled: map.loadStatus === Enums.LoadStatusLoaded
    onClicked: {
      var point = ArcGISRuntimeEnvironment.createObject("Point", {
        x: -157.564,
        y: 20.677,
        spatialReference: SpatialReference.createWgs84()
      });
      mv.setViewpointCenterAndScale(point, 4000000.0);
    }
  }
}
```

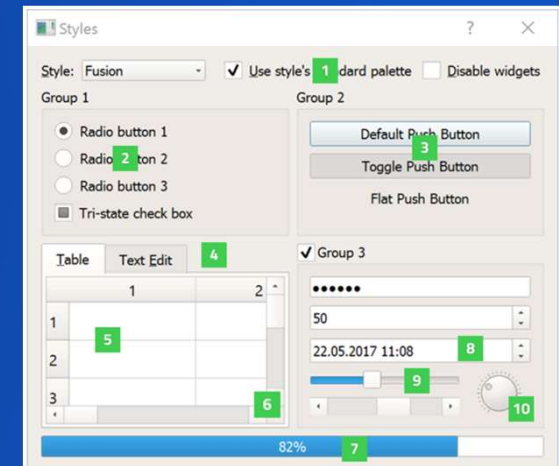
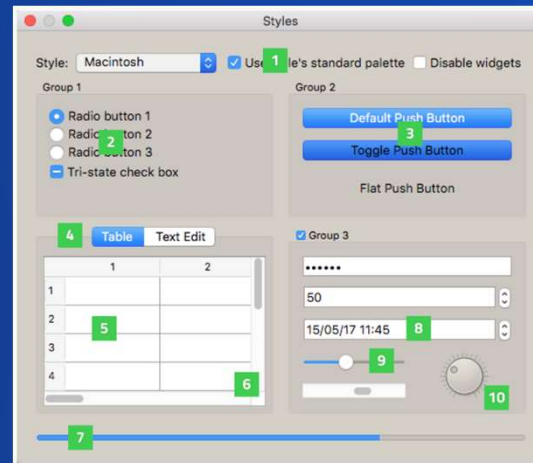
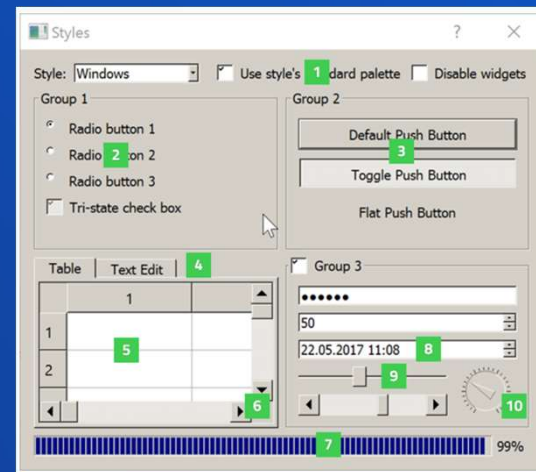
ArcGIS Runtime

Dynamic property binding

# Qt Widgets (Desktop)

## Styled look and feel

- Available Styles
  - Windows style
  - Mac style
  - Fusion style





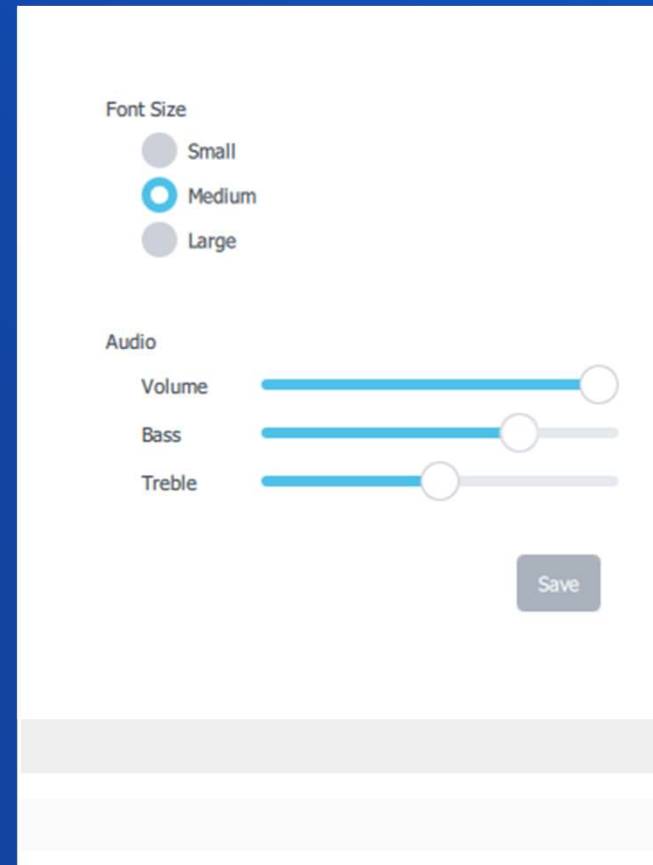
# Qt Quick Controls (QML – all platforms)

Styled look and feel

- Available Styles

- Default style
- Material style
  - Google's guidelines
- Universal style
  - Microsoft's guidelines
- Fusion style
  - Desktop-oriented look and feel
- Imagine style
  - Based on image assets

- **Platform style**



# Qt Demo

Cross platform sensors



# What are the Pros and Cons?

ArcGIS Runtime SDK for Qt

## Pros

- Write once
- Same APIs and code
- Consistent, styled look and feel
- Access to device sensors
- Open-source community
- QML or C++ - based on experience

## Cons

- Niche Native APIs unavailable through Qt (e.g. AirPlay)
- Won't match the look and feel of the native platform (but can be a pro)
- Qt framework can increase apps size
- Setup can take some time



**.NET**

Rich Zwaap



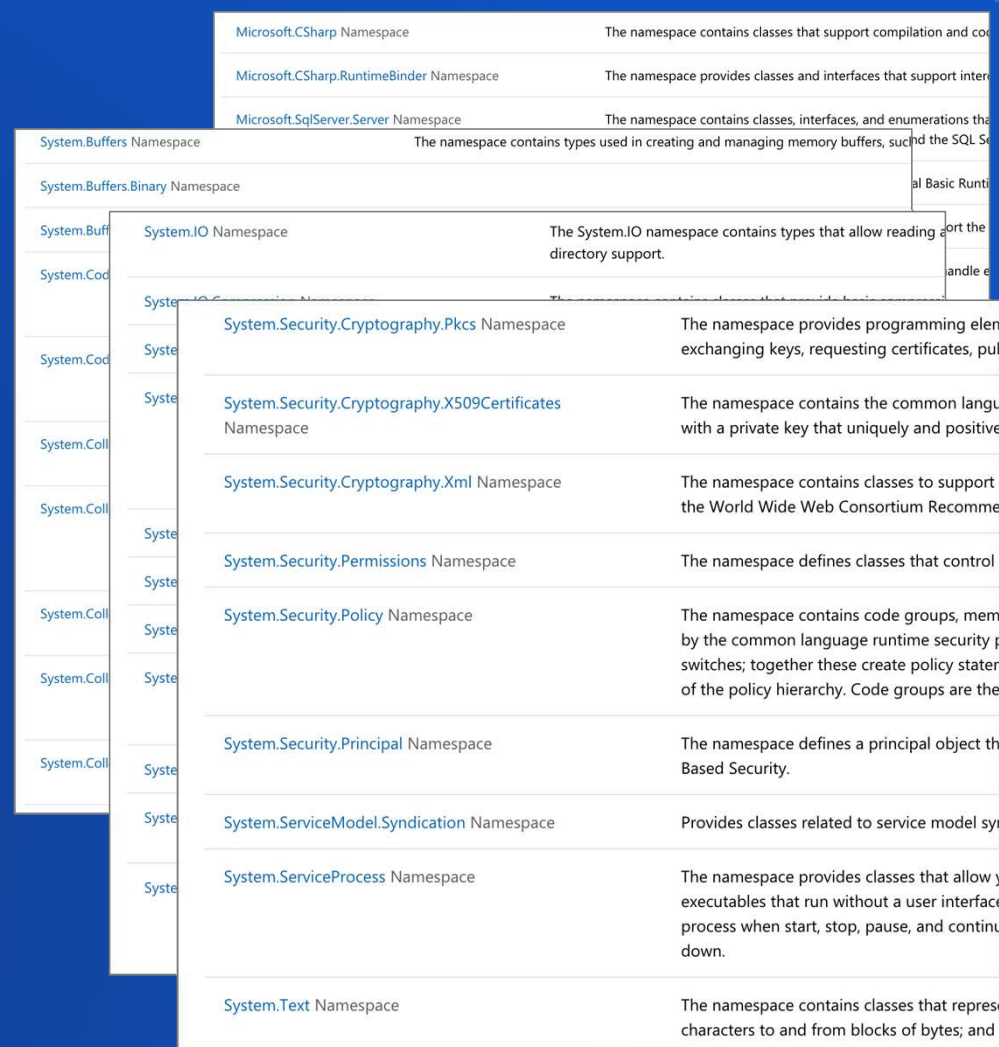
# Agenda - .NET and Xamarin

- Xamarin
- ArcGIS Runtime for .NET
- Demo
- Pros and Cons



# What is Xamarin?

- MS dev for iOS and Android
- Four main parts
- 1) .NET on iOS and Android
  - Base Class Library
  - Primitives, collections, data objects, IO, networking, reflection, exception handling, logging, etc, etc
  - Same across platforms
  - *Shared business logic*



Microsoft.CSharp Namespace	The namespace contains classes that support compilation and code generation.
Microsoft.CSharp.RuntimeBinder Namespace	The namespace provides classes and interfaces that support interop between C# and dynamic languages.
Microsoft.SqlServer.Server Namespace	The namespace contains classes, interfaces, and enumerations that support the SQL Server CLR integration.
System.Buffers Namespace	The namespace contains types used in creating and managing memory buffers, such as the SQL Server CLR integration.
System.Buffers.Binary Namespace	The namespace contains types used in creating and managing memory buffers, such as the SQL Server CLR integration.
System.Buffers.Text Namespace	The namespace contains types used in creating and managing memory buffers, such as the SQL Server CLR integration.
System.IO Namespace	The System.IO namespace contains types that allow reading and writing files and directories.
System.Security.Cryptography.Pkcs Namespace	The namespace provides programming elements for public key infrastructure (PKI) operations, such as exchanging keys, requesting certificates, pulling certificates, and validating certificates.
System.Security.Cryptography.X509Certificates Namespace	The namespace contains the common language runtime (CLR) classes for X.509 certificates, including a private key that uniquely and positively identifies the certificate.
System.Security.Cryptography.Xml Namespace	The namespace contains classes to support the XML Signature and XML Encryption specifications of the World Wide Web Consortium Recommendation.
System.Security.Permissions Namespace	The namespace defines classes that control access to resources.
System.Security.Policy Namespace	The namespace contains code groups, membership rules, and policy statements. Code groups are used to identify assemblies and assemblies are associated with code groups. Membership rules are used to determine the policy state of a code group. Policy statements are used to determine the policy state of a code group.
System.Security.Principal Namespace	The namespace defines a principal object that represents the current user and a role-based security system.
System.ServiceModel.Syndication Namespace	Provides classes related to service model syndication.
System.ServiceProcess Namespace	The namespace provides classes that allow you to create and manage services that run without a user interface. Services are processes that run in the background and can be started, stopped, paused, and continued.
System.Text Namespace	The namespace contains classes that represent characters and provide methods for converting characters to and from blocks of bytes; and

# What is Xamarin?

- 2) Native iOS and Android APIs...
  - Just exposed in C# and Visual Studio
  - Same APIs used by native iOS and Android devs
  - UI, sensors, notifications, background tasks, app architecture, etc
  - *Full power of each platform*

The image shows a screenshot of the Android Studio documentation interface. At the top, there is a section for **UIStackView**, described as a streamlined interface for laying out a collection of views in either a column or a row. It lists SDKs: iOS 9.0+, tvOS 9.0+, and Framework: UIKit. Below this is the **Declaration** section showing the code: `class UIStackView : UIView`. The main part of the screenshot is for the **RelativeLayout** class, which is noted as being added in API level 1. It shows the class hierarchy: `public class RelativeLayout extends ViewGroup`, with subclasses `android.view.View`, `android.view.ViewGroup`, and `android.widget.RelativeLayout`. It also lists known direct subclasses: `DialerFilter` and `TwoLineListItem`. A description follows: "A Layout where the positions of the children can be described in relation to each other or to the parent." A note states: "Note that you cannot have a circular dependency between the size of the RelativeLayout and the position of its children. For example, you cannot have a RelativeLayout whose height is set to `WRAP_CONTENT` and a child set to `ALIGN_PARENT_BOTTOM`." A final note mentions a measurement bug in platform version 17 and lower, affecting child views' `MeasureSpec` values, with a link to `MeasureSpec.makeMeasureSpec` for more details.

# What is Xamarin?

- 3) Abstraction libraries
  - Xamarin Forms for UI
  - Xamarin Essentials for everything else
  - UWP (in addition to Android and iOS)
  - Enable code-sharing outside of core business logic
- 4) Tooling
  - Fully featured IDE support
  - Debugging, design surface, profiling, intellisense, test instrumentation, etc

## Get Started with Xamarin.Essentials

Follow the [getting started guide](#) to install the **Xamarin.Essentials** NuGet package into your existing or new Xamarin.Forms, Android, iOS, or UWP projects.

## Feature Guides

Follow the guides to integrate these Xamarin.Essentials features into your applications:

- [Accelerometer](#) – Retrieve acceleration data of the device in three dimensional space.
- [App Information](#) – Find out information about the application.
- [Barometer](#) – Monitor the barometer for pressure changes.
- [Battery](#) – Easily detect battery level, source, and state.
- [Clipboard](#) – Quickly and easily set or read text on the clipboard.
- [Color Converters](#) – Helper methods for System.Drawing.Color.
- [Compass](#) – Monitor compass for changes.
- [Connectivity](#) – Check connectivity state and detect changes.
- [Detect Shake](#) – Detect a shake movement of the device.
- [Device Display Information](#) – Get the device's screen metrics and orientation.
- [Device Information](#) – Find out about the device with ease.
- [Email](#) – Easily send email messages.
- [File System Helpers](#) – Easily save files to app data.
- [Flashlight](#) – A simple way to turn the flashlight on/off.
- [Geocoding](#) – Geocode and reverse geocode addresses and coordinates.
- [Geolocation](#) – Retrieve the device's GPS location.
- [Gyroscope](#) – Track rotation around the device's three primary axes.
- [Launcher](#) – Enables an application to open a URI by the system.



# What is the ArcGIS Runtime SDK for .NET?

- Surfaces all Runtime capabilities to .NET devs
- Supports:
  - Windows Presentation Foundation (WPF)
  - Universal Windows Platform (UWP)
  - Xamarin.Android
  - Xamarin.iOS
  - Xamarin.Forms (Android, iOS, and UWP)
  - .NET Standard 2.0\*
- Same non-UI API surface for all platforms
- Native platform and Xamarin Forms UI components

# Demo

.NET, UWP, and Xamarin



# .NET and Xamarin – Pros and Cons

- Pros

- Xamarin and .NET are (mostly) open source
- Target all platforms in a single IDE, on a single OS
- Full access to all native platform APIs
  - Updates delivered in sync with underlying platforms
- Platform abstractions also available
- Large and active community



# .NET and Xamarin – Pros and Cons

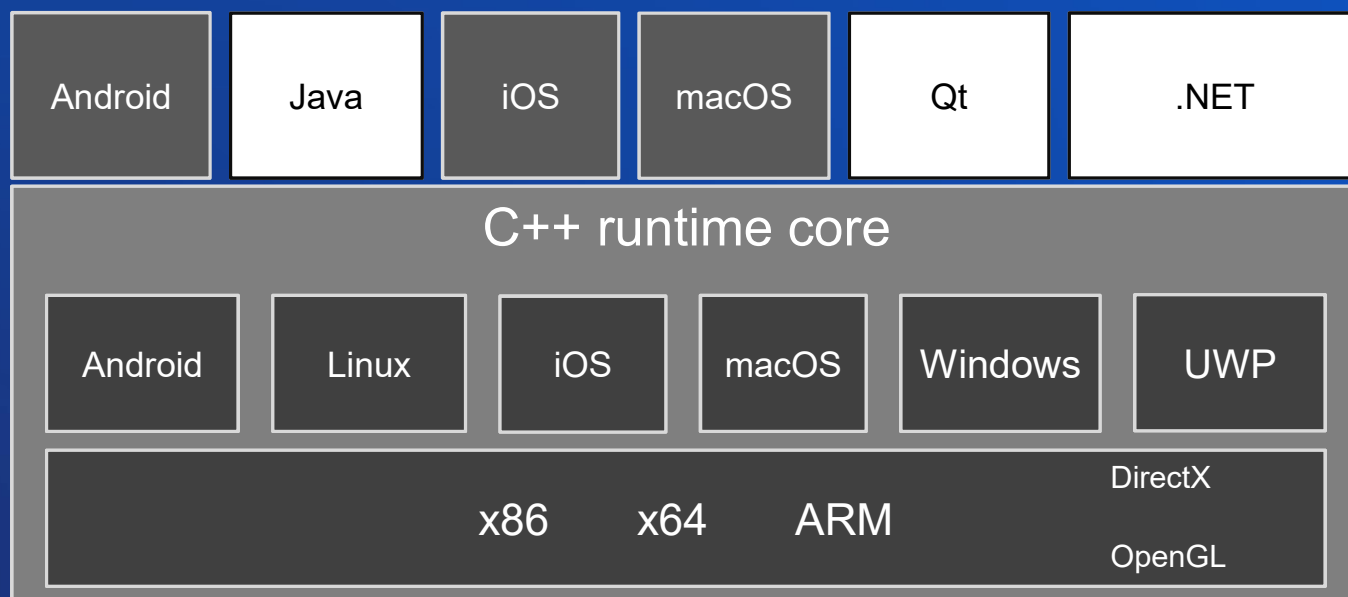
- Cons

- Visual Studio not free for most commercial uses
- Not 100% abstraction of all platform code
- Significant technology stack (Visual Studio, Android SDKs, XCode...)
- Very rapid release cadence
- You need a Mac and a Windows machine to target all platforms



# ArcGIS Runtime cross-platform options

- All Runtime APIs built on common Runtime core



# Summary

- How do you choose a cross platform SDK?
  - Understand expectations of your users
  - Understand skillset of your developers
- Options and ArcGIS Runtime SDKs
  - Java               => ArcGIS Runtime SDK for Java
  - Qt                   => ArcGIS Runtime SDK for Qt
  - .NET/Xamarin      => ArcGIS Runtime SDK for .NET



# Other sessions

- *Wednesday, July 10*

- **ArcGIS Runtime SDK for Qt: Building Apps**  
12:15 PM – 1 PM  
Demo Theater 8

- *Thursday, July 11*

- **ArcGIS Runtime SDK for Java: Building Apps**  
10:30 AM – 11:30 AM  
Demo Theater 8
- **ArcGIS Runtime SDKs: Building .NET Apps**  
2:30 PM – 3:30 PM  
Room 30 D

*Thursday, July 11*

**ArcGIS Runtime:  
Road Ahead**

8:30 AM – 9:30 AM  
Room 15 B

**ArcGIS Runtime:  
Building Offline Applications**

2:30 PM – 3:30 PM  
Room 31 A



# Questions?

[developers.arcgis.com/arcgis-runtime](https://developers.arcgis.com/arcgis-runtime)

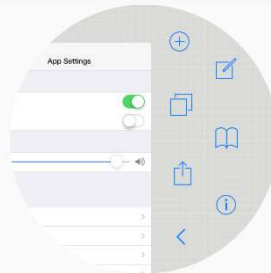
## What are ArcGIS Runtime SDKs?

Built natively from the ground up using C++ and GPU acceleration, ArcGIS Runtime SDKs expose the full capability of the ArcGIS Platform to mobile, desktop, and embedded devices. Whether you're using ArcGIS Online or ArcGIS Enterprise or have disconnected users, ArcGIS Runtime SDKs let you do all things GIS, from simple map display or routing to advanced analysis.

Choosing the Right Esri API



Work Offline



Native User Experience

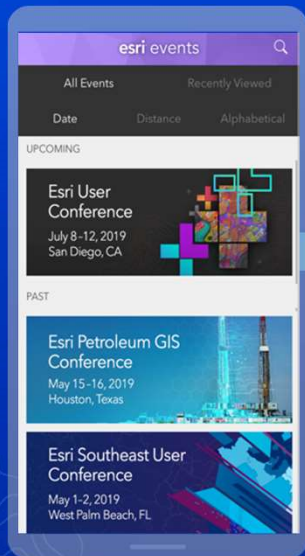


Access Native APIs

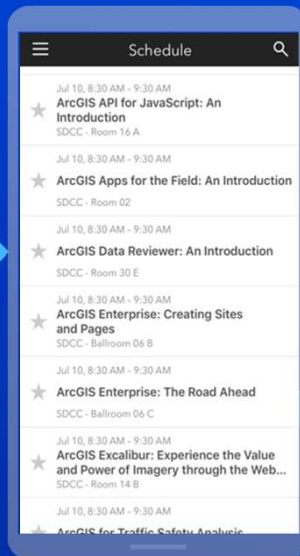


# Please Share Your Feedback in the App

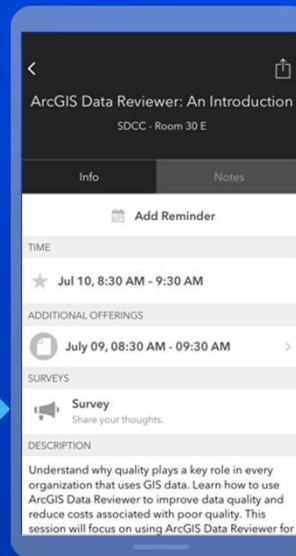
Download the Esri Events app and find your event



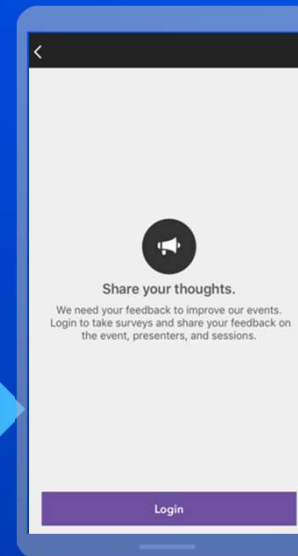
Select the session you attended



Scroll down to "Survey"



Log in to access the survey



Complete the survey and select "Submit"

